A* Decoding of Block Codes

L. Ekroot S. Dolinar

Jet Propulsion Laboratory, California Institute of Technology

June 20, 1995

Abstract— *Tho* **A* algorithm is applied to maximum-likelihood soft-decision decoding of binary linear block codes. This paper gives a tutorial on the A* algorithm, compares the decoding complexity with that of exhaustive search and Viterbi decoding algorithms, and presents performance curves obtained for several codes.**

Keywords— **maximum likelihood decoding, soft-decision decoding, binary linear block codes, algorithm A*.**

I. Introduction

The A* algorithm is an artificial intelligence tree-search algorithm for finding the path in a graph that optimizes a function defined over all paths. Nilsson [1] describes the algorithm as a heuristic graph-search procedure, and shows that the algorithm always terminates in an optimal path. A* has been used to implement full maximum likelihood soft decoding of linear block codes by Han, et al, [2], [3], and [4]. Other tree-search algorithms, e.g., Stack, Fano, and M-, do not result in maximum likelihood decoding.

This paper describes the fundamentals of the A* algorithm as it is applied to maximum likelihood decoding of binary linear block codes. For this work, A* was used in decoding simulations for several codes. This resulted in comparisons of the complexity of A* decoding to that of other maximum likelihood decoding methods, and accurate word error rate curves.

Binary symbols, $b_i \in \{0,1\}$, from an (n,k) linear code are transmitted using binary antipodal signaling, i.e., $c_i = (-1)^{b_i}$, over an additive white Gaussian noise channel. The received symbols, r_i, are continuous valued *soft* symbols. The hard-limited *symbol* h_i *is* the transmitted signal value, ± 1, nearest to the received symbol r_i.

The research reported in this paper was carried out in the Communications Systems Research Section of the Jet Propulsion Laboratory, California Institute of Technology under a contract to the National Aeronautics and Space Administration.

A. Maximum likelihood decoding

Maximum likelihood soft-decision decoding decodes a received sequence to the codeword c* that maximizes the likelihood of the received soft symbols. It is convenient to think of the codewords, which are length N sequences of ± 1 's, and the received sequence $\boldsymbol{r}$ as points in N-dimensional space. Assuming an additive white Gaussian noise channel, the codeword c* that maximizes the likelihood of the received sequence $\boldsymbol{r}$ is the one that minimizes the Euclidean distance between the received word $\boldsymbol{r}$ and t] ie codeword $\boldsymbol{c}$.

The codeword that is closest to the received word can be found by exhaustively checking all possible codewords, or by cleverly seeking out the one that minimizes the distance. For an (N, K) code, there arc 2^K codewords to check, making an exhaustive search prohibitive for most interesting codes. Viterbi decoding the block code on a trellis can accomplish maximum likelihood decoding more efficiently, using a smaller fixed number of calculations [6], and [7]. Techniques such as A* that use a heuristic search to find the maximum likelihood codeword can significantly reduce the average number of calculations needed for decoding, especially at high SNR.

B. Linear codes as trees

Define C to be an (N, K) linear code with 2^K length N binary codewords b $\in \mathcal{C}$. The *generator matrix* G for the code is a K x N matrix of zeros and ones whose rows are linearly independent codewords. Given K information bits in a row vector $\boldsymbol{x}$, the corresponding binary codeword is b $= \boldsymbol{x}G$. If G is in *systematic* form, the K information bits are directly visible in the codewords. For the codes considered 1 iere, the first K columns of G form an identity matrix, and the codewords can be divided into information bits, in the first K positions, and parity bits, in the last N -- K positions.

To apply a heuristic graph search algorithm to the decoding of a block code, the code is thought, of as a binary tree with 2^K depth N leaves where each path from the root to a leaf corresponds to a codeword. If the codewords of a systematic code live in a binary tree,

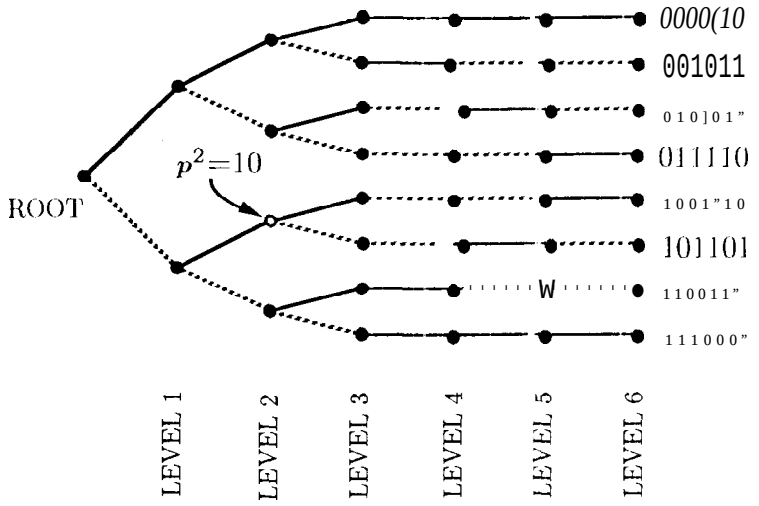


Fig. 1. Binary tree representation, of the (6,3) shortened Hamming code.

through level K – 1 the tree is full, that is, every node has two descendants. This is because any length K sequence of zeros and ones can be an information sequence. Since the parity bits arc determined by the information bits, every level K node has only one descendant path which continues to level N. Any node of level $1< K$ in the tree is fully defined by the path $p^l = p_1 p_2 \ldots p_l$ of information bits p_i from the root to that node.

Example: Consider the (6,3) shortened Hamming code with the generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Figure 1 shows the representation of this code as a tree with solid and dashed edges used to represent zeros and ones respectively, or equivalently, transmitted +1s and −1s. Each node above level 3 has two child nodes, while each node at levels 3, 4, or 5 has only one descendant. The leaf nodes at level 6 represent the eight codewords of the code. The highlighted node at level 2 illustrates the path labeling.

1 I. ALGORITHM DESCRIPTION

The A* algorithm searches a graph for the path that minimizes a *path metric* function. On any given iteration, it expands the node that is likely to yield the optimal path, and eliminates any nodes that can only have suboptimal descendants. The method by which nodes are selected for expansion and eliminated from consideration uses an underestimate of the path metric function, called a heuristic function. The heuristic function at anode must lower bound the true path metric function, for all paths that pass through that node.

For maximum likelihood soft-decision decoding of an (N, K) block code received over the additive white Gaussian noise channel, the path metric function is the square of the Euclidean distance between a codeword and the received word:

$$s(\boldsymbol{r}, \mathrm{c}) = \sum_{i=1}^{N} (r_i - c_i)^2$$

For the algorithm to find the maximum likelihood codeword, the value of the heuristic function at a node must be less than or equal to the actual squared distance for any full-length path that passes through that node, i.e., between the received word and any codeword that is prefixed by $\boldsymbol{p}^l$, the path that defines the node.

The minimum squared distance over all codewords that begin with the path $\boldsymbol{p}^l$ is lower bounded by the minimum squared distance over all length N binary sequences that begin with the path $\boldsymbol{p}^l$, i.e.,

$$\min_{\{\boldsymbol{b} \in \{0,1\}^N \mid b_i = p_i,\ i=1,\ldots,l\}} s(\boldsymbol{r}, \boldsymbol{c}) \leq \min_{\{\boldsymbol{b} \in \mathcal{C} \mid b_i = p_i,\ i=1,\ldots,l\}} s(\boldsymbol{r}, \boldsymbol{c})$$

where $c_i = (-1)^{b_i}$. The minimum squared distance over all length N sequences that begin with the path $\boldsymbol{p}^l$ is achieved by the sequence that begins with $\boldsymbol{p}^l$ and continues with binary symbols consistent with the hard-limited received symbols. Thus, a, valid heuristic function for this problem is the squared distance from the received sequence to either the codeword defined by the path $\boldsymbol{p}^K$, "if the node is at level l = K, or the sequence that begins with the

path $\boldsymbol{p}^l$ and is completed by symbols consistent with the hard-limited symbols, if 1< K. For 1< *K, 1* lan, et al, [2] use a heuristic function that minimizes $s(\boldsymbol{r}, \mathrm{c})$ over just those length N sequences that have legitimate weights in the code rather than over all length N sequences as we have done in this work.

A. Fundamentals of the algorithm

The A* algorithm maintains an ordered list of nodes. Associated with each node is the path $\boldsymbol{p}^l$ that identifies the node, the value of the heuristic function, and an indicator of whether the node represents a single codeword. The values of the heuristic function determine the order of the nodes on the list and therefore guide the search through the tree. The algorithm expands any node that might yield a codeword with the minimum distance from the soft received symbols, and eliminates all nodes that are too far from the received symbols to have the maximum likelihood codeword as a possible descendant.

When the algorithm begins the search, the root of the tree is tile only node on the list. At each iteration, the node on the top of the list, which has the smallest value of the heuristic function, is expanded. It is taken off the list and the two possible ways of continuing the path are considered as nodes to put back on the list. Each new node is placed back on the list provided that its heuristic function value is not greater than the actual path metric for a completed codeword. If the node expanded is at level K — 1, the two level K children specify codewords, and the value of the heuristic function at each child node is the actual squared distance between the codeword and the received word. These codewords are called *candidate codewords.* When a node that defines a codeword is placed back on the list, all nodes below it are deleted. The algorithm terminates when a candidate codeword reaches the top of the list. That codeword is the maximum likelihood codeword.

B. Features that improve the efficiency

The two features described in this section are not necessary to guarantee maximum likelihood soft-decision decoding, but they improve the algorithm's efficiency. Han, et al, [2] proposed sorting the bit positions according to the reliability of the received symbols to reduce the average number of nodes expanded. Our implementation incorporates this sorting feature, and additionally introduces a simplification to the heuristic function which reduces the number of computations during each node expansion.

A. Sorting by reliability

If the bit positions corresponding to the more reliable received symbols are expanded first, then the search will be directed more quickly to close candidate codewords. The nearer a symbol is to 0, the less reliable it is because it is almost equally far from both +1 and – 1. It follows that the greater the magnitude of the received symbol the more reliable that symbol is. To take advantage of the most reliable symbols first, the received symbols are reordered in descending order by magnitude, and the code symbols are reordered equivalently. Reordering the code symbols is equivalent to reordering the columns of the generator matrix.

This implementation of the A* algorithm sorts the received symbols by reliability, reorders the columns of the generator matrix in the same way, and then tries to row reduce the generator matrix so that it is systematic. However, if it encounters a column, among the first K columns, that is linearly dependent on previous columns, it moves the offending column and corresponding received symbol to the end before proceeding. Of course, the code and received symbols are kept in the same order as the columns of the generator matrix.

Typically the number of nodes expanded while decoding a received word is significantly reduced by sorting the symbols before starting the decoding process. For the shorter codes like the (24, 12) Golay code which could be tested both with and without sorting, the increase in decoding efficiency from sorting the symbols was found empirically to outweigh the cost

of sorting and row reducing the generator matrix. For the larger codes, such as the (48,24) quadratic residue code, decoding without sorting was so much more time consuming that it was not a reasonable option to run comparison tests. Sorting was adopted as a standard feature.

B. Sign-Magnitude Path Metric Function

Every soft symbol r_i is at least as far away from the codeword symbol c_i as it is from the had-limited symbol h_i. The squad distance, $s(\boldsymbol{r}, \mathrm{c})$, can be written as the sum of the square of the distance to the hard-limited symbols, $s(\boldsymbol{r}, \boldsymbol{h})$, and an amount $a(\boldsymbol{r}, \mathrm{c})$ that is nonzero only when at least one symbol c_i c1ocs not equal the corresponding hard-limited symbol h_i, as follows:

$$
\begin{aligned}
s(\boldsymbol{r}, \boldsymbol{c}) &= \sum_{i=1}^{N} (r_i - c_i)^2 \\
&= \sum_{\substack{i=1 \\ h_i = c_i}}^{N} (r_i - h_i)^2 + \sum_{\substack{i=1 \\ h_i \neq c_i}}^{N} (r_i + h_i)^2 \\
&= \sum_{i=1}^{N} (r_i - h_i)^2 + \sum_{\substack{i=1 \\ h_i \neq c_i}}^{N} 4 h_i r_i \\
&= \sum_{i=1}^{N} (r_i - h_i)^2 + 4 \sum_{\substack{i=1 \\ h_i \neq c_i}}^{N} |r_i| \qquad (1) \\
&= s(\boldsymbol{r}, \boldsymbol{h}) + 4a(\boldsymbol{r}, \boldsymbol{c}) \qquad (2)
\end{aligned}
$$

where (1) uses $h_i r_i$ ‘ $\mathrm{sgn}(r_i) r_i = |r_i|$, and (2) introduces the sign-magnitude path metric function

$$
a(\boldsymbol{r}, \mathrm{c}) = \sum_{\substack{i=1 \\ \mathrm{sgn} r_i \neq c_i}}^{N} |r_i|
$$

as an alternative to the squared Euclidean distance. Since the first term in (2) does not depend on the codeword c, it is constant over the minimization, and

$$\min_{\boldsymbol{c}} s(\boldsymbol{r}, \boldsymbol{c}) = s(\boldsymbol{r}, \boldsymbol{h}) + 4 \min_{\boldsymbol{c}} a(\boldsymbol{r}, \mathrm{c}).$$

Maximum likelihood decoding of the received sequence can be done by finding the codeword that minimizes either $s(\boldsymbol{r}, \mathrm{c})$ or $a(\boldsymbol{r}, \mathrm{c})$.

Because each term of $a(\boldsymbol{r}, \mathrm{c})$ is either zero or $|r_i|$ based on a comparison, it is simpler to calculate than $s(\boldsymbol{r}, \boldsymbol{c})$, which for each i requires a difference and a square. Because of this simplification, the sign-magnitude path metric has found application in Viterbi decoder implementations such as the high performance Viterbi decoder developed at the Jet Propulsion Laboratory for the Deep Space Network [8]. For the A* application, it has the additional advantage of allowing the heuristic function to be independent of future symbols deeper in the tree. The heuristic function at level $l < \mathrm{K}$ is simply the accumulated path metric through level l, because zero additional metric is contributed if the path is completed by symbols consistent with the hard-limited symbols.

Example revisited: Consider the (6,3) shortened Hamming code and the received sequence $\boldsymbol{r}$ = (.05, -1.3, 1.1, .8, -.25, .6). Reordering the received vector by reliability gives $\boldsymbol{r}'$=(-1.3, 1.1, .8, .6, -.25, .05). The reordered generator matrix in systematic form is

$$G' = \begin{pmatrix} 1\ 0\ 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ ]\quad 0\ 1\ 1 \end{pmatrix}.$$

Figure 2 shows the tree explored by the A* algorithm when the sorted code is used. Each node is labeled with the value at that node of the heuristic function using the sign-magnitude path metric. The 3 expanded nodes are each designated by a o; the 2 candidate codewords are each designated by a A; and the only edges shown in the figure are the 12 edges explored before the algorithm terminates. The nodes with paths O, 11 and 101 are dropped from the list when the candidate word, with path metric .65, is put on the list. The search

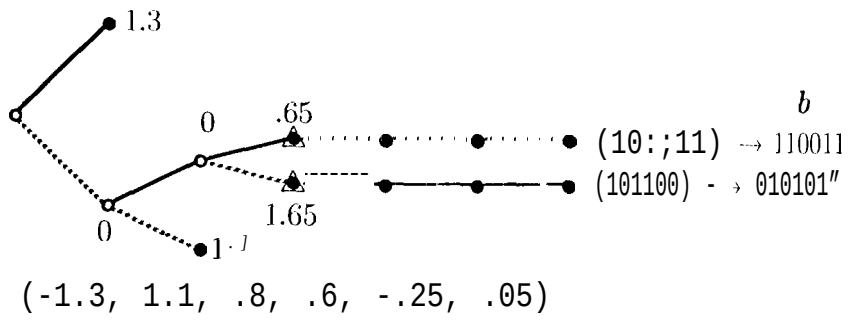


Fig. 2. The tree explored by the A* algorithm when the bit positions are sorted to take advantage of the more reliable symbols first. Expanded nodes are designated by o, and nodes defining candidate codewords are designated by Δ.

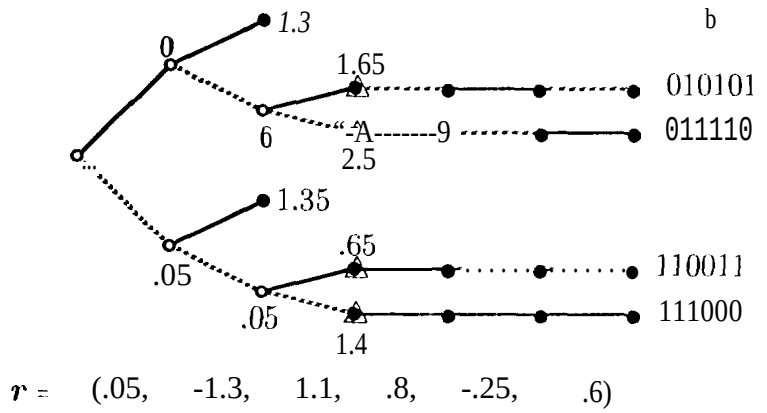


Fig. 3. The tree explored by the A* algorithm when the bit positions are not sorted Expanded nodes are designated by o, and nodes defining candidate codewords are designated by Δ.

prompt] y terminates because the top node on the list defines a candidate codeword, namely $b' = 100111$. Unshuffling b' gives the maximum likelihood decoded codeword in the original symbol order, $b = 110011$. For comparison Figure 3 shows the larger tree explored by the algorithm when the symbols are not sorted.

C. Verification of the decoder

The decoding results of the A* algorithm were compared to the results of two exhaustive search decoder implementations. 'The (24,12) Golay code was used for this test since it has

only $2^{12} = 4096$ codewords making it feasible to get timely results from an exhaustive search decoder. First, the software decoded the received sequences using both A* and exhaustive search, and compared the results internally. Second, a couple hundred received sequences were decoded by both the A* software and an independent exhaustive search decoder written in APL. The results showed that both exhaustive search and A * decoders decoded the same noisy vectors to the same codewords.

The software to implement the A* algorithm has been written in C and run on several Sun platforms. Since integers on these processors are 32 bits long, the software to implement the A* algorithm has been constrained to linear codes with 64 or fewer bits per codeword by using two 32 bit integers for each codeword. Because of this implementation detail, it was important to confirm that the A* software properly decodes codes longer than length 32. Most interesting codes with lengths over 32 bits take a prohibitively long time to decode exhaustively. A test code with length N greater than 32, and one with more than 32 information bits were devised so they could be readily decoded by other means. The code with length greater than 32 was created by repeating the parity bits of the (24,12) Golay code. This formed a (36,12) code that was no more difficult to exhaustively decode than the (24, 12) Golay code. After debugging and testing, the decoder decoded 500 codewords consistent with the exhaustive decoder results. Next a simple (34,33) code, consisting of 33 information bits and 1 overall parity bit, was tested on two hundred noisy received words. This code was selected because a maximum likelihood decoder is easy to write, and an APL program was used to verify that the two hundred test words decoded consistently.

D. Operational details

To analyze the performance of either the algorithm or a code, data were taken by running the software with different input. parameters. For a given run, the software can take as input the generator matrix for the code, the SNR, the seed for the random number generator, and the number of words to decode. It returns the average number of nodes expanded,

the average number of candidate codewords, and the number Of word errors that occurred. Sometimes a system call from inside the program was used to provide the amount of CPU time consumed during a run. These decoding runs ranged in size from hundreds to tens of thousands of decoded received sequences. The codes that have been examined include a (63,56) Bose-Chaudhuri-Hocquenghem (BCH) code, a (48,24) quadratic residue code, a (24,12) Golay code, a (31,10) BCH code, and a (32,16) Reed-Muller code. The data from multiple runs were combined carefully to give the results in the following sections.

III. A* Algorithm Complexity

The intricacy of the A* algorithm makes it difficult to count the exact number of calculations necessary to decode a received sequence. The algorithm's complexity can be roughly measured by various indicators of the size of its search tree. These include the number of candidate codewords, the number of expanded nodes, and the number of edges searched in the tree. The number of edges E in the search tree is given by

$$E = 2X + (N - K)C, \tag{3}$$

where X is the number of nodes expanded including the root, and C is the number of candidate codewords. Because the search size for the A* algorithm varies from one received sequence to the next, the averages of these numbers over many received sequences are used for comparison. The next two sections present simulation results demonstrating that the average search size and the average time to decode arc related linearly, and showing how the average size of the A* search tree varies with the signal-to-noise ratio.

A. *Time to decode versus search size*

The average amount of time it takes to decode received sequences reflects both the computational overhead for each sequence decoded and the computation is for each part of the search tree. Analyzing the time to decode requires that all of the timing data be taken on

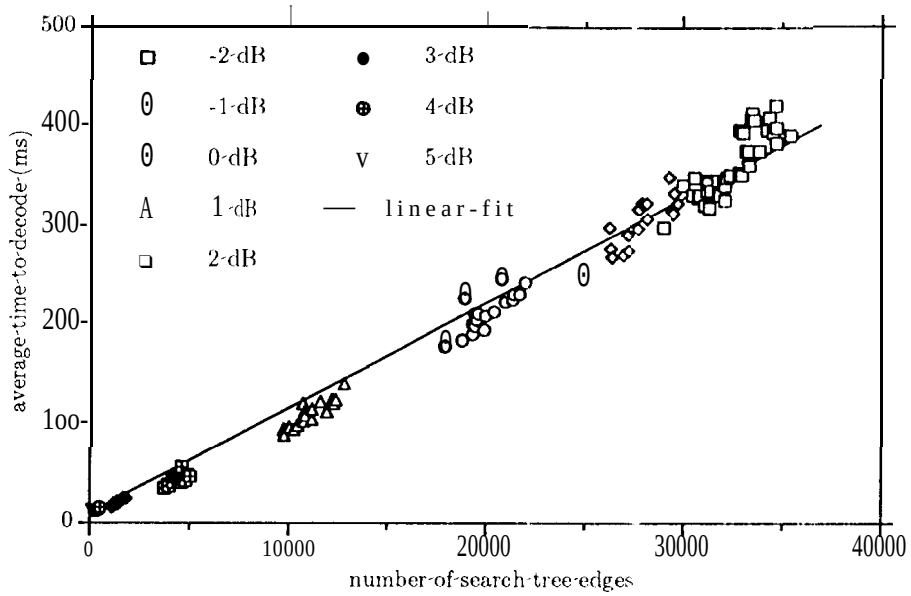


Fig. 4. Scatter plots of average CPU time per decoded word versus average number of search tree edges per decoded word for the (48,24) quadratic residue code on a Sparc 10 Model 30 workstation.

the same computer, and that the accuracy of the timing data be sufficient to perform comparisons. The system call used to generate the timing information for a run was accurate to within a second, which is too coarse to study data on individual decoded sequences, but sufficient for data on ensembles of decoded sequences.

The relationship between the indicators of search size introduced earlier and decoding time may be observed in the data from many runs for the (48,24) quadratic residue code on a Spare 10 Model 30 workstation. The average decoding time versus the average number of search tree edges is shown in Figure 4 along with a weighted linear fit [1] to the data. Although the data displays a small amount of statistical variability, the time to decode displays a nearly linear relationship to the indicator of search size.

[1] The number of decoded words in each run was included in the line fitting process to account for the variation in accuracy between data from large and small runs.

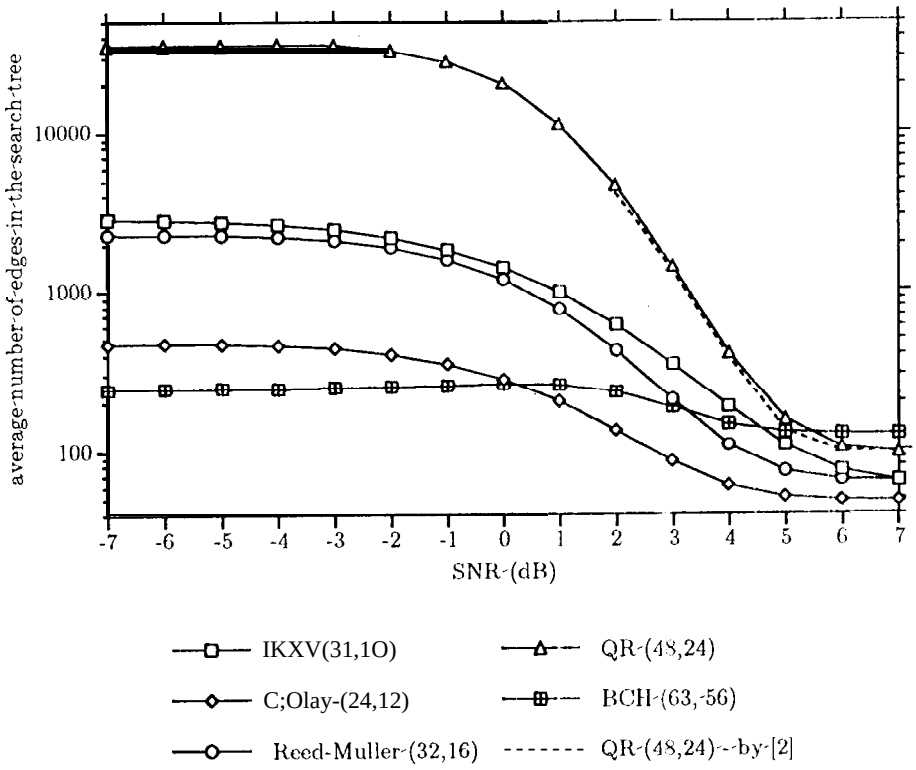


Fig. 5. The average number of edges in the search tree for several codes as a function of SNR

B. Search size versus SNR

The average size of the tree that A* searches is a function of the SNR for the received sequences. For each of the codes studied, the average number of search tree edges is shown versus SNR in Figure 5. Also shown for the QR(48,24) code is the average of edges searched by Han, et al, computed by applying (3) to the average of nodes and candidates reported in [2].

Not surprisingly, for extremely high SNR, the A* algorithm typically finds only two candidate words, along the way expands K nodes, and therefore has a search tree with $E = 2K + 2(N - K) = 2N$ edges. For low SNR, the soft symbols are predominantly noise,

but the A* algorithm still expands a mere fraction of the nodes in the tree, especially as it bases early decisions on the symbols that contribute most to the final choice. Figure 5 shows that the average number of edges is almost constant for SNR below – 4 dB. The algorithm was also tested for each code with no signal at all, i.e., an SNR of $-\infty$ dB; the complexity measures for this case were found to be consistent, with the limiting values in Figure 5.

IV. Comparisions with Other Maximum Likelihood Decoders

Many other maximum likelihood soft-decision decoding algorithms use a fixed number of calculations to decode any received sequence independent of SNR. This section compares the fixed decoding complexity of some of these decoders with the average decoding complexity of the A* algorithm.

A. Exhaustive search and full tree search

An exhaustive search decoder calculates the path metric for each codeword individually, and returns the codeword with the minimum metric. For an (N, K) Mock code, an exhaustive search decoder must compute the path metrics for all 2^K codewords. If exhaustive search is cast in terms of a graph with one edge for each bit in each codeword, the number of edges for an exhaustive search is $N2^K$, independent of the SNR.

A slightly more efficient technique to compute the path metrics for all the codewords is to use the full code tree. Here the path metric for a codeword at a leaf is the sum along the path to that leaf of the branch metrics associated with each edge. For an (N, K) code, the number of edges in the full tree is $(N - K + 2)2^K - 2$. This technique checks all 2^K leaves, but has fewer edges than an exhaustive search.

B. Viterbi decoding of block codes

To apply soft-decision Viterbi decoding to a block code the code is represented as a trellis. Bahl, et al [9], Wolf [10] and Massey [11] introduce a minimal trellis for decoding block codes. McEliece [6] shows a simple technique for constructing, the minimal trellis for a given code,

and also shows that it is optimal for Viterbi decoding complexity. A Viterbi decoder for a code on a trellis uses a constant number of calculations and comparisons independent of signal-to-noise ratio. The Viterbi decoding complexity can be measured by the total number of edges in the trellis.

An (N, K) code has a minimal trellis that can be constructed from the generator matrix. Different permutations of a code may have different minimal trellises. For many nice codes, such as cyclic codes, the minimal trellis has more edges than the minimal trellises for other permutations of the code. The permutation that gives the most edges is the worst permutation of the code. The number of edges in the minimal trellis for the worst permutation is no more than $(N - 2A4 + 4)2^M - 4$ where $M = \min(K, N - K + 1)$. Other permutations can give smaller minimal trellises.

C. *Comparisons of Decoding Complexity*

The search size for the A * algorithm depends on the received sequence, and the average search size depends on S N R. The averages and standard deviations of the number of edges searched when no signal is present can be used to compare the complexity of A* decoding with that of maximum likelihood decoders that use a fixed number of calculations.

Table 1 shows for each code the number of edges used for an exhaustive search, for the full code tree, and for Viterbi decoding on minimal trellises corresponding to the worst and best code permutations (where known). All of these are much greater than the average number of edges in the A* search tree, shown in the table for the two limiting cases when SNR is $\pm\infty$ dB. The standard deviations of the numbers of edges searched by A* at $-\infty$ dB are substantial, but still less than the averages in all codes tested.

Consider for example the (24, 12) Golay code. An exhaustive search explores $24 \cdot 212 = 98304$ edges. The full tree has $14.212 - 2 = 57342$ edges. The number of edges in the minimal trellis for the worst permutation of the (24, 12) Golay code is 4.212--4: 16380. The number of edges in the minimal trellis for the best permutation is 3580 [6]. By comparison the A*

Number of edges for various maximum likelihood decoders	BCH (31, 10)	Golay (24, 12)	Reed-Muller (32, 16)	Quadratic residue (48, 24)	BCH (63, 56)
A* algorithm search tree at $+\infty$ dB (average $\pm$ standard dev.)	62 ± 0	48 ± 0	64 ± 0	96 ± 0	126 ± 0
A* algorithm search tree at $-\infty$ dB (average $\pm$ standard dev.)	2903 ± 1660	469 ± 327	2303 ± 1912	34,429 $\pm 31,940$	245 ± 151
Minimal trellis for the best code permutation (fixed)	≤ 7068 ≥ 3484	3580	6396	860,156	≤ 5068 ≥ 4892
'Minimal trellis for the–worst code permutation (fixed)	15,356	16,380	262,140	67,108,860	13,052
Full code tree (fixed)	23,550	57,342	1,179,646	436,207,614	6.49×10^{17}
Exhaustive search (fixed)	31,744	98,304	2,097,152	805,306,368	4.54×10^{18}

Table 1. Comparisons of decoding complexity for some maximum likelihood decoding techniques

algorithm searches an average of $469 + 327\ (1\sigma)$ edges in the low-SNR limit, and $48 \pm O\ (1\sigma)$ edges in the high-SNR limit.

For a larger code such as the (48,24) quadratic residue code, the edge counts for an exhaustive search and the full code tree are astronomical. The minimal trellis for a worst permutation still has an impractical number, $4 \cdot 2^{24} - 4 = 67108860$, of edges. Recent work [12] and [13] has produced this code's best permutation which results in a minimal trellis with 860156 edges. By comparison the A* algorithm's search tree has on average $34429 \pm 31940\ (1\sigma)$ edges in the low-SNR limit and $96 \pm O\ (1\sigma)$ edges in the high-SNR limit.

The total number of edges explored by each of these maximum likelihood decoding algorithms equals the number of binary additions that must be performed in order to compute the required path metrics. It is possible for special algorithms to reduce the number of computations slightly for some special codes. For example, by using enhancements on a certain trellis for the Golay code, Forney [7, p. 1183] can perform maximum likelihood decoding with a total of 1351 binary operations, of which 840 are binary additions/subtractions and 511

are binary comparisons. The number of additions/subtractions for this special construction is still greater than the? A* algorithm's average plus one standard deviation in the low-SNR limit.

V. Code Performance

The A* algorithm that we have implemented has been very useful for simulating code performance. Figure 6 shows the probability of word error versus SNR for the (63,56) BCH, (32,16) Reed-Muller, (31,10) BCH, (24,12) Golay, and (48,24) quadratic residue codes. The error bars are one standard deviation of an average of m independent Bernoulli trials. Specifically, the estimated standard deviation is $\sigma = \sqrt{\frac{p(1-p)}{m}}$, where p is the estimate of the probability of word error at a given SNR, and m is the number of words decoded at that SNR.

VI. Enhancements of the Decoding Algorithm

This paper has described an implementation of the A* decoding algorithm that searches a comparable average number of edges to the algorithm reported in ['2]. This is illustrated in Figure 5. 'J'here are many directions in which the algorithm might be enhanced. Here we briefly discuss some of them,

A. Improved Stopping Rule Based on the Code Geometry

If the angle between the received word and a candidate codeword, as viewed from the origin, is less than half the minimum angle between any two codewords, then that candidate codeword must be the closest one to the received word. One of the suggestions in [2] is to calculate the angle between the received word and each candidate codeword as it is found. If this angle guarantees that the candidate is the closest codeword, then declare that the decoding is complete, and exit the algorithm.

The bounded angle decoding condition is very easy to check when the A* decoder uses the

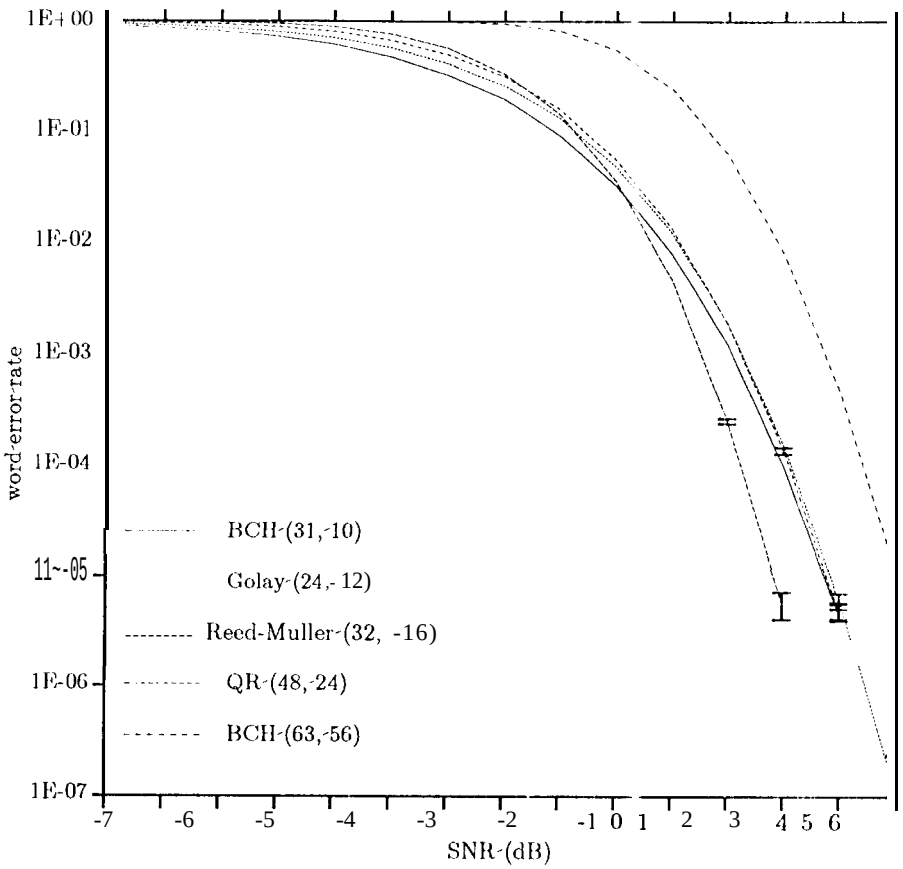


Fig. 6. Probability of word error versus SNR for the (63,56) BCH, (32,16) Reed-Muller, (31 ,10) BCH, (24,12) Golay, and (4 S,24) quadratic residue codes. The error bars are $\pm\sigma$, one standard deviation.

sign-magnitude path metric function $a(\boldsymbol{r}, c)$. The received word $\boldsymbol{r}$ falls within the bounded angle decoding region of the codeword $\boldsymbol{c}$ if and only if

$$a(\boldsymbol{r},\ c\ ) \frac{1}{2}\left[\sum_{i=1}^{N}|r_i| - \sqrt{(N-d)\sum_{i=1}^{N} r_i^2}\right] \triangleq A(\boldsymbol{r}) \tag{4}$$

where d_{min} is the minimum Hamming distance of the code. The decoder's stopping rule can be modified to terminate whenever a candidate codeword either reaches the top of the

ordered list of nodes or has a sign-magnitude path metric less than the threshold $A(r)$. This feature was not implemented for the results presented in this paper, but it is expected to reduce the size of the tree searched.

B. Applying the Best Code Permutation

Each parity bit is a linear function of a subset of the information bits. If a node in the tree is deep enough to specify all the information bits for a particular parity bit, then any codeword passing through that node will have the same value for that parity bit. The heuristic function could use this parity bit to improve the distance underestimate for all nodes at that depth, and thus reduce the size of the tree searched by the algorithm.

An equivalent way to accomplish the same result without requiring the heuristic function to look ahead is to permute a parity column of the generator matrix to immediately follow the last information column on which it depends. A node at this level of the tree is not expanded to two successors, but extends to one unique descendant, which may expand with the next information bit. Some of these column permutations may be obtained for free, simply by not moving any linearly dependent columns discovered while row reducing the generator matrix after sorting its columns by symbol reliability.

The theory of minimal trellises and the implementation of A* decoding are both concerned with finding efficient code permutations. For minimal trellises, this is a static problem with a static solution: the structure of the code dictates which permutations are good and which are not. For A* decoding the best permutation also depends on the value of the received symbols. For the codes we have tested, sorting by reliability appears to have more powerful consequences on reducing the average size of the search tree than optimally ordering the symbols a priori. Other codes, such as block-truncated convolutional codes, possess much static structure that would be severely disturbed by any type of sorting unrelated to their natural trellis structure; for such codes the best A* permutation is probably closer to the best a priori permutation than to the ordering according to symbol reliability.

For any given code and set of received symbols, the algorithm could try to determine an optimum permutation of code symbols that produces the most efficient decoding. Of course, if the decoder were to devote considerable resources to this task, it In i ght negate any extra efficiency obtained.

C. Using Orderly Node Extensions as Well as Expansions

In the current version of the algorithm, all expansions of nodes are performed one at a time. One node is expanded into two nodes, the heuristic function is evaluated at both new nodes, and the ordered list of nodes is updated and consulted before the next node is expanded. 1 lowever, when a node at level K is reached, it is extended to level N by adding its final $N - K$ branch metrics all at once. An algorithm that performs its extensions in an orderly manner, one at a time, will have a search tree with the same number of expanded nodes and fewer edges (or at most the same number).

Extending nodes one at a time require.s additional updating and checking of the ordered list of nodes, so the overall decoding algorithm may not be as efficient. The choice of which technique is better will depend on the particular code. For example, orderly extensions are probably more effective for low rate codes than for high rate codes.

VII. Conclusions

The application of the A* algorithm to maximum-likelihood soft-decision decoding allows for efficient simulation of code performance. 'l'he A* algorithm fi Ids the codeword that maximizes the like] i hood of the received word given the codeword. This is equivalent to minimizing either the Euclidean distance between the received word and a codeword or the al ternative sign-magn itude path metric. The use of a heuristic function constrains the search to only a subtree of the code's finite binary tree. The heuristic function underestimates the true path metric function in order to assure that the subtree contains the optimal path.

The size of the tree searched by the A* algorithm, as described by the total number of

its edges, is a good indicator of the complexity for decoding that received sequence. Since the search size depends on the received sequence, the average search size as a function of signal-to-noise ratio is used for comparison. The search tree is smallest for high SNR where the algorithm goes straight to the maximum-likelihood codeword, and larger at low SNR where the searched portion of the tree is still much smaller than the full code tree. At low SNR the average size of the A* search tree is also smaller than the best possible fixed trellis size for Viterbi decoding.

For many codes, maximum likelihood decoder error rates can be estimated by simulations using the A* algorithm, whereas such tests would he impractical with other methods. For research applications, these simulations are useful for comparisons to theoretical bounds, and for testing other predictors of code performance.

ACKNOWLEDGMENTS

The authors would like to thank A. Kiely, R. McEliece, F. Pollara, and L. Swanson for many helpful discussions.

REFERENCES

[1] N. J. Nilsson, *Principles of Artificial Intelligence,* Palo Alto , CA: Tioga Publishing Co., 1980.

[2] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, *Efficient Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes Using Algorithm A*,* Technical Report SU-CIS-91-42, School *of Computer and Information Science,* Syracuse *University, Syracuse,* NY *13244,* December 1991.

[3] Y. S. Han, and C.R. J'. Hartmann, Designing Efficient Maximum-Likelihood Soft-Decision Decoding Algorithms for Linear Block Codes Using Algorithm A*, Technical Report SU-CIS-92-10, *School of Computer and Information Science,* Syracuse *University, Syracuse, NY 1.?244,* June 1992.

[4] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, Efficient Priority-First Search Maximum-Likelihood So ft-Decision Decoding of Linear Block Codes, to appear in IEEE Transact ions on Information Theory, September 1993.

[5] S. Benedetto, E. Biglieri, V. Castellani, *Digital Transmission Theory,* Englewood Cliffs, N.]: Prentice-Hall, Inc., 1987.

[6] R. J. McEliece, The Viterbi Decoding Complexity of Linear Block Codes, presented *as a long paper at IEEE* 1S17''94, Trondheim, Norway.

[7] G. D. Forney, Jr., Coset Codes - Part 11: Binary Lattices and Related Codes. *IEEE Trans. Inform. Theory,* vol. IT-34 (1988), pp. 1152-1187.

[8] J. Statman, G. Zimmerman, F. Pollara, and O. Collins, A Long Constraint Length VLSI Viterbi Decoder for the DSN, *TDA Progress Report 42-95* July - September 1988, Jet Propulsion Laboratory, Pasadena, California, November 15, 1988.

[9] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate, *IEEE Trans. Inform. Theory,* vol. IT-20 (1974), pp. 284- 287.

[10] J. K. Wolf, Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inform. Theory,* vol. 1'1'-24 (1978), pp. 76-80.

[11] J. L. Massey, Foundations and methods of channel coding. *Proc. Int. Conf. Inform. Theory and Systems*, NTG-Fachberichte vol. 65, Berlin, September 18-20, 1978, pp. 148-157.
[12] S. Dolinar, J. Ekroot, A. Kiely, W. Lin, and R. J. McEliece, "The Permutation Trellis Complexity of Linear Block Codes," *Proc. 32nd Annual Allerton Conference on Communication, Control, and Computing*, Allerton, Illinois, October 1994.
[13] A. Kiely, S. Dolinar, R. McEliece, J. Ekroot, W. Lin, Trellis Decoding Complexity of Linear Block Codes submitted to *IEEE Trans. Inform. Theory*.

A* Decoding of Block Codes

L. Ekroot S. Dolinar

June 20, 1995

Abstract— **The A* algorithm is applied to maximum-likelihood soft-decision decoding of binary linear block codes. This paper gives a tutorial on the A* algorithm, compares the decoding complexity with that of exhaustive search and Viterbi decoding algorithms, and presents performance curves obtained for several codes.**

Keywords— **maximum likelihood decoding, soft-decision decoding, binary linear block codes, algorithm A*.**

I. Introduction

The A* algorithm is an artificial intelligence tree-search algorithm for finding the path in a graph that optimizes a function defined over all paths. Nilsson [1] describes the algorithm as a heuristic graph-search procedure, and shows that the algorithm always terminates in an optimal path. A* has been used to implement full maximum likelihood soft decoding of linear block codes by Han, et al, [2], [3], and [4]. Other tree-search algorithms, e.g., Stack, Fano, and M-, do not result in maximum likelihood decoding.

This paper describes the fundamentals of the A* algorithm as it is applied to maximum likelihood decoding of binary linear block codes. For this work, A* was used in decoding simulations for several codes, This resulted in comparisons of the complexity of A* decoding to that of other maximum likelihood decoding methods, and accurate word error rate curves.

Binary symbols, $b_i \in \{0, 1\}$, from an (n,k) linear code arc transmitted using binary antipodal signaling, i.e., $c_i = (-1)^{b_i}$, over an additive white Gaussian noise channel. The received symbols, r_i, are continuous valued *soft* symbols. The *hard- limited symbol hi* is the transmitted signal value, ± 1, nearest to the received symbol r_i.

The research reported in this paper wss carried out in the Communications Systems Research Section of the Jet Propulsion Laboratory, California Institute of Technology under a contract to the National Aeronautics and Space Administration,

To apply a heuristic graph search decoding algorithm, the code is associated with a systematic generator matrix G and represented by a corresponding binary tree with 2^k depth N leaves representing codewords, Any node at level $l \leq k$ in the tree is fully defined by the path $\boldsymbol{p}^l = p_1 p_2 \ldots p_l$ of information bits p_i from the root to that node.

II. Algorithm Description

The A* algorithm searches the code tree for the path that minimizes a *path metric* function. On each iteration, it expands a node that is likely to yield the optimal path, and eliminates any nodes that can only have suboptimal descendants, Nodes are selected for expansion and eliminated from consideration using an underestimate of the path metric function, called a heuristic function. The heuristic function at a node must lower bound the true path metric function, for all paths that pass through that node.

A. Fundamentals of the Algorithm

The A* algorithm maintains a list of nodes ordered by their heuristic function values. When the algorithm begins the search, the root of the tree is the only node on the list. At each iteration, the node on the top of the list, which has the smallest value of the heuristic function, is expanded into two new nodes. Each new node is placed back on the list provided that its heuristic function value is not greater than the actual path metric for a completed codeword. If the expanded node is at level *k --1,* the two level *k* children specify *candidate codewords,* and the value of the heuristic function at each child node is the path metric for the specified codeword. When a node that defines a codeword is placed back on the list, all nodes below it are deleted. The algorithm terminates when a candidate codeword reaches the top of the list. That codeword is the maximum likelihood codeword.

B. Choice of Path Metric and Heuristic Functions

For maximum likelihood soft-decision decoding of an (n, k) block code received over the additive white Gaussian noise channel, the path metric function is the square of the Euclidean distance between the received word $\boldsymbol{r}$ and a codeword c, namely, $s(\boldsymbol{r}, \mathrm{c}) = \sum_{i=1}^{n} (r_a - c_i)^2$.

The minimum path metric over all codewords that begin with the path $\boldsymbol{p}^l$ is lower bounded by the minimum path metric over all length N binary sequences that begin with the path $\boldsymbol{p}^l$. The latter is achieved by the sequence that begins with $\boldsymbol{p}^l$ and continues with binary symbols consistent with the hard-limited received symbols. Thus, a valid heuristic function for this problem is the path metric from the received sequence to either the codeword defined by the path $\boldsymbol{p}^k$, if the node is at level $l = k$, or the sequence that begins with the path $\boldsymbol{p}^l$ and is completed by symbols consistent with the hard-limited symbols, if $l < k$.

In our implementation we have substituted for the Euclidean distance the equivalent sign-magnitude path metric function

$$a(\boldsymbol{r}, \boldsymbol{c}) = \sum_{\substack{i=1 \\ \mathrm{sgn} r_i \neq c_i}}^{n} |r_i| = \frac{1}{4}[s(\boldsymbol{r}, \boldsymbol{c}) - s(\boldsymbol{r}, \boldsymbol{h})],$$

where $\boldsymbol{h}$ is the vector of hard-limited symbols corresponding to $\boldsymbol{r}$. Because each term of $a(\boldsymbol{r}, c)$ is either zero or $|r_i|$ based on a comparison, it is simpler to calculate than $s(\boldsymbol{r}, \mathrm{c})$, which for each i requires a difference and a square. For the A* application, $a(\boldsymbol{r}, \mathrm{c})$ has the additional advantage of allowing the heuristic function to be independent of future symbols deeper in the tree: at any node in the tree, zero additional metric is contributed if the path is completed by symbols consistent with the hard-limited symbols.

Han, et al, [2] use the path metric $s(\boldsymbol{r}, \mathrm{c})$ and a slightly tighter heuristic function that minimizes $s(\boldsymbol{r}, \mathrm{c})$ over just those length n sequences that have legitimate weights in the code rather than over all length n sequences as we have done in this work. However, we observed in our tests (e.g., see Figure 2 below) that the simpler heuristic function could be substituted with only a minor effect on average decoding complexity.

C. *Received Symbol Sorting*

Han, et al, [2] noted that, if the bit positions corresponding to the more reliable received symbols are expanded first, then the search will be directed more quickly to close candidate codewords. The greater the magnitude of the received symbol the more reliable that symbol is. Our implementation of the A* algorithm also sorts the received symbols by reliability,

because decoding without sorting was found to be much more time consuming. The algorithm reorders the columns of the generator matrix to match the symbol sorting, and then tries to row reduce the generator matrix so that it is systematic. If it encounters a column, among the first k columns, that is linearly dependent on previous columns, it moves the offending column and corresponding received symbol to the end before proceeding

Example: Consider the (6,3) shortened Hamming code and the received sequence r = (.05, -1.3, 1.1, .8, -.25, .6). Reordering the received vector by reliability gives r' = (-1.3, 1.1, .8, .6, - .25, .05). The original and reordered generator matrices in systematic form are $G = \begin{pmatrix} 1&0&0&1&1&0 \\ 0&1&0&1&0&1 \\ 0&0&1&0&1&1 \end{pmatrix}$ and $G' = \begin{pmatrix} 1&0&0&1&1&1 \\ 0&1&0&1&1&0 \\ 0&0&1&0&1&1 \end{pmatrix}$. Figure la shows the tree explored by the A* algorithm when the sorted code is used. Each node is labeled with the value at that node of the heuristic function using the signal-magnitude path metric. There are 3 expanded nodes, 2 candidate codewords, and only 12 edges are explored before the algorithm terminates. The nodes with paths 0, 11 and 101 are dropped from the list when the candidate word, with path metric .65, is put on the list. The search promptly terminates because the top node on the list defines a candidate codeword, namely $b' = 100111$. Unshuffling b' gives the maximum likelihood decoded codeword in the original symbol order, $b = 110011$. For comparison Figure lb shows the larger tree explored by the algorithm when the symbols are not sorted.

III. A* Algorithm Complexity

The intricacy of the A* algorithm makes it difficult to count the exact number of calculations necessary to decode a received sequence. We found that the time to decode is to first order proportional to the number of edges in the search tree, and therefore we have used the number of search tree edges E as a rough measure of the algorithm's complexity. Because the search size for the A* algorithm varies from one received sequence to the next, both averages and standard deviations of E arc used for comparison with that of maximum likelihood decoders that usc a fixed number of calculations.

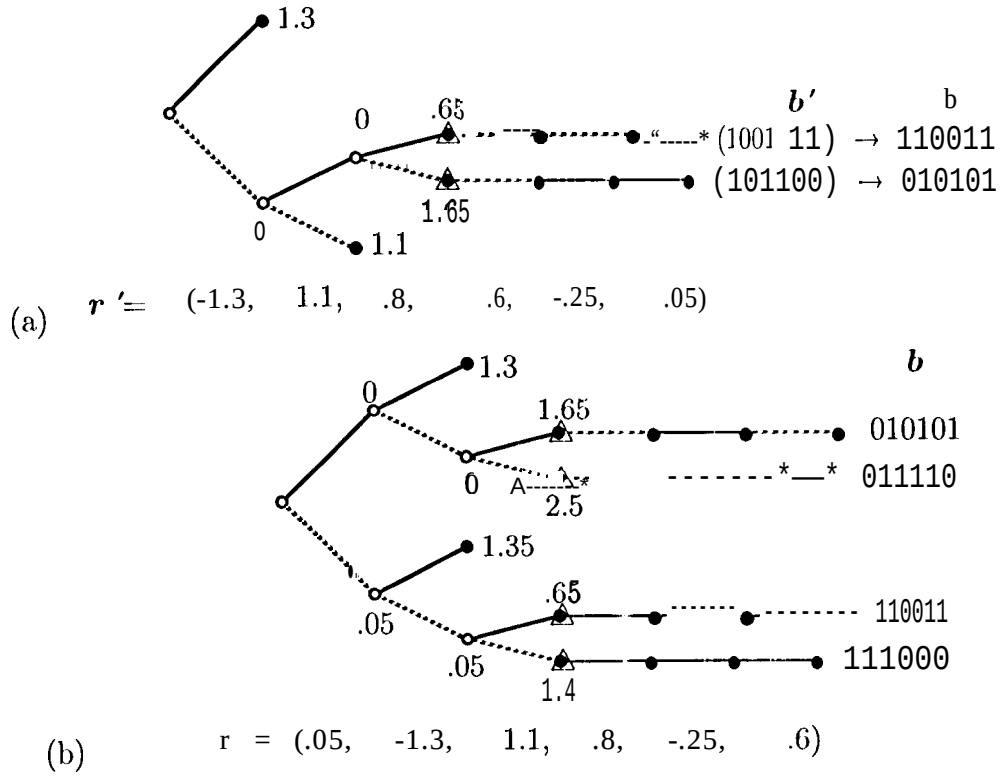


Fig. 1. (a) The tree explored by the A* algorithm when the bit positions are sorted to take advantage of the more reliable symbols first. (b) The tree explored by the A* algorithm when the bit positions are not sorted. Expanded nodes predesignated bye, and nodes defining candidate codewords are designated by A. Code symbols 0 and 1 are represented by solid and dashed edges, respectively.

*A. A * Search Size Versus Signal-to-Noise Ratio*

The average size of the tree that A* searches is a function of the signal-to-noise ratio $\mathcal{E}_b/N_0$ for the received sequences. For each of the codes studied, the average number of search tree edges per code symbol, E/n, is shown versus $\mathcal{E}_b/N_0$ in Figure 2.

For extremely high $\mathcal{E}_b/N_0$, the A* algorithm typically finds only two candidate words, along the way expands k nodes, and therefore has a search tree with $E = 2k + 2(n-k) = 2n$ edges which is seen in Fig. 2. For very low $\mathcal{E}_b/N_0$, the A* algorithm still expands only a small fraction of the nodes in the full tree, because it bases its early decisions on those symbols that contribute most to the final choice. Even though all the symbols are predominantly noise in

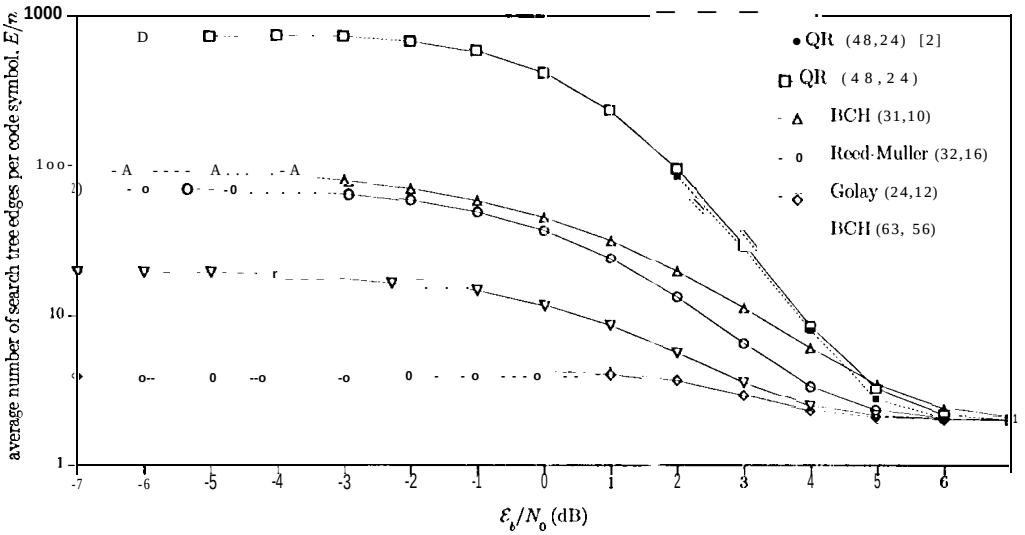


Fig. 2. The average number of search tree edges per code symbol for several codes as a function of sig,,rd-to-noise ratio, Also shown for the QR(48,24) code is the average of edges per code symbol searched by Han, et al, computed from $2X + (n-k)C$ where X is the average number of nodes expanded, and C is the average number of candidate codewords reported in [2],

this case, the algorithm is relatively efficient in focusin ıg its attention on the symbols that predominate the maximum likelihood decision. As $\mathcal{E}_b/N_0$ goes to -- ∞ dB, the average number of edges seems to approach a constant value for each code, which (somewhat surprisingly) can be slightly lower than the worst-case average occur ring at a nonzero $\mathcal{E}_b/N_0$.

B. *comparisons with other Maximum Likelihood Decoding Methods*

Many other maximum likelihood soft-decision decoding algorithms explore a fixed number of edges E in a search tree or trellis.[1] For an exhaustive search decoder $E = n2^k$; for a slightly more efficient full tree search, $E = (n-k+2)2^k - 2$.

It has been noted [5] that efficient soft-decision decoding of block codes may be accomplished by applying the Viterbi algorithm to a minimal trellis representation of the code. The

[1] For each of these algorithms, the decoding complexity is dominated by the number of edges examined, even though other operations are involved. For A* these other operations include sorting the symbols and row reducing the generator matrix, and determining where to insert newly expanded nodes on the list. When the sign-magnitude path metric is used, half of the node insertions are trivial, because the metric of one of the two expanded nodes is the same as that of its parent node.

Viterbi decoding complexity can be measured by the total number of edges in the trellis [6]. However, different permutations of a code may have different minimal trellises with different numbers of edges. The worst permutation of the code has no more than $(n - 2m + 4)2^m - 4$ edges, where $m = \min(k, n - k + 1)$. Determining the best permutation is an area of active research (e.g., [7], [8]).

Table 1 shows for each code studied the number of edges used for an exhaustive search, and for Viterbi decoding on minimal trellises corresponding to the worst and best code permutations. All of these are much greater than the average number of edges in the A* search tree, shown in the table for the two limiting cases when $\mathcal{E}_b/N_0$ is $\pm\infty$ dB. The standard deviations of the numbers of edges searched by A* at · -cm dB arc very large, but they still indicate (e. g., with the aid of Chebyshev's inequality) that A* almost always searches many fewer edges than the other methods,[2] except possibly for the BCH (31 ,10) code, the lowest rate code in the table.

IV. Code Performance

For many codes, maximum likelihood decoder erroi rates can be estimated by simulations using the A* algorithm, whereas such tests would be impractical with other methods. Figure 3 shows simulated word error rate versus $\mathcal{E}_b/N_0$, found using A* for the same codes shown in Figure 2.

Acknowledgments

The authors would like to thank A. Kiely, F. Pollara, and especially R. McEliece for many helpful discussions. The referees' comments were also exceptionally valuable.

Note: An extended version of this paper originally appeared in [10].

[2] For hardware implementations, the maximum number of edges searched by A* should also be considered. However, the worst-case maximum is of dubious significance, because it is always possible (though extremely unlikely) to contrive a set of Gaussian noise samples that force A* to explore every edge in the full code tree, Practical implementations can be designed to achieve near-maximum-likelihood performance based on studying the full histogram of the number of edges explored, or, more crudely, the mean and variance as we have reported here.

Number of edges for various maximum likelihood decoders	BCH (31, 10)	Golay (24, ii?)	Reed-Muller (32, 16)	Quadratic residue (48, 24)	BCH (63, 56)
A* algorithm search tree at $+\infty$ dB (average ± standard dev.)	62 ± 0	48 ± 0	64 ± 0	96 ± 0	126 ± O
A* algorithm search tree at $-\infty$ dB (average ± standard dev.)	2903 ±1660	469 ±327	2303 ±1912	34,429 ±31,940	245 ±151
Minimal trellis for the best code permutation (fixed)	≤ 7068 >3484.	3580	6396	860,156	≤ 5068 ≥ 4892
Minimal trellis for the worst code pm-mutation (fixed)	15,356	16,380	262,140	67,108,860	13,052
Full code tree (fixed)	23,550	57,342	1)179,646	436,207,614	6.49 x 10^{7}
Exhaustive search (fixed)	31,744	98,304	2,097,152	805,306,368	4.54X101*

Table 1. Comparisons of decoding complexity for some maximum likelihood decoding techniques. The edge counts for the best permutations are from a table in [9], or from applying bounds and techniques reported there.

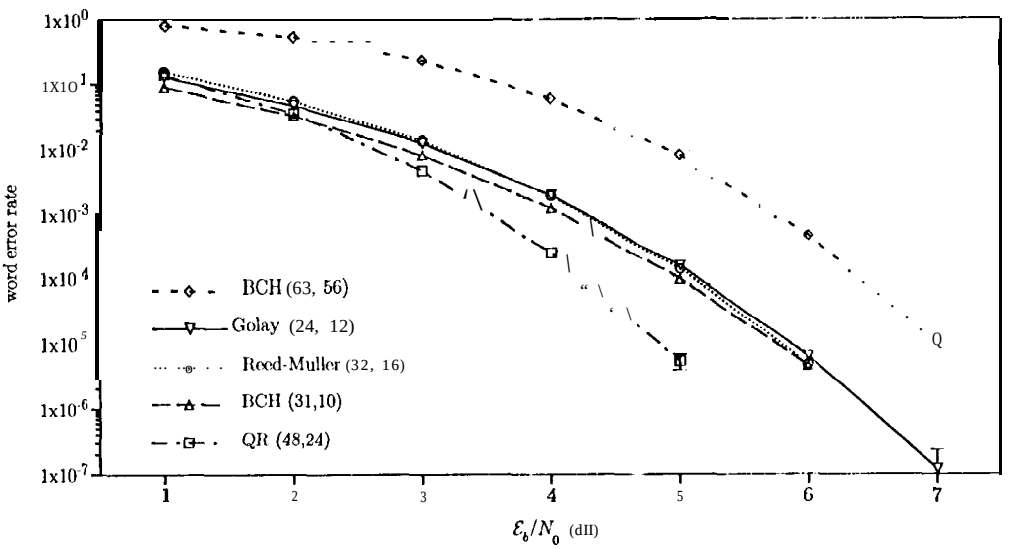


Fig. 3. Simulated word error rate versus $\mathcal{E}_b/N_0$. The error bars are ∃ one standard deviation, and they are only shown where they are larger than the plot symbol.

REFERENCES

[1] N. J. Nilsson, *Principles of Artificial Intelligence,* Palo Alto , CA: Tioga Publishing Co., 1980.

[2] Y. S. Han, C.R.P. Hartmann, and C.-C. Chen, "Efficient Maximum. Likelihood Soft-I) ecision Decoding of Linear Block Codes Using Algorithm A*," Technical Report SU-CIS-91-42, *School of Computer and Information Science, Syracuse University, Syracuse, NY 13244,* December, 1991.

[3] Y. S. Han, and C. R.P. Hartmann, "Designing Efficient Maximum-Likelihood Soft-I)ecision Decoding Algorithms for Linear Block Codes Using Algorithm A*," Technical Report SU-CIS-92-10, *School of Computer and Information Science, Syracuse University, Syracuse, NY 13244,* June, 1992.

[4] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient Priority-First Search Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes," *IEEE Trans. Inform. Theory,* vol. IT-39, pp. 1514-1523, September, 1993.

[5] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for MinimizingSymbol Error Rate," *IEEE Trans. Inform. Theory,* vol. IT-20, pp. 284-287, March, 1974.

[6] R. J. McEliece, "The Viterbi Decoding Complexity of Linear Block Codes," *presented as* along paper at *IEEE ISIT'94,* Trondheim, Norway.

[7] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the Optimum Bit Orders with Respect to the State Complexity of Trellis Diagrams for Binary Linear Codes," *IEEE Trans. Inform. Theory,* vol. IT-39, pp. 242-245, January, 1993.

[8] A. Kiely, S. Dolinar, R. McEliece, L. Ekroot, W. Lin, "Trellis Decoding Complexity of Linear Block Codas," submitted to *IEEE Trans. inform. Theory.*

[9] S. Dolinar, L. Ekroot, A. Kiely, W. Lin, and R. J. McEliece, "The Permutation Trellis Complexity of Linear Block Codes," *Proc. 32nd Annual Allerton Conference on Communication, Control, and Computing,* Allerton, Illinois, October, 1994.

[10] L. Ekroot and S. Dolinar, "Maximum-Likelihood Soft-Decision Decoding of Block Codes Using the A* Algorithm," *TDA Progress Report 42-117* January - March 1994, Jet Propulsion Laboratory, Pasadena, California, pp.)29-144, May 15, 1994.

2. CO_2 Removal Technology Trade

Six different technologies were included, as shown in Figure V-7 for wet weight and Figure V-8 for electrical power demand. These figures illustrate the impact of technology substitution on the various subsystems and the entire system: while a technology candidate can show significant weight or power advantages over other candidates at that functional level (e.g., CO_2 removal) , the advantage may not be maintained through the subsystem (e.g., air revitalization) and through the entire life support system.

The wet weights of various systems considered for comparison of CO_2 removal technologies could differ by as much as 340 kg (for the 90-day mission) primarily due to differences in the demand for stored supplies and in the weight of process equipment. Differences in process equipment weights for the various CO_2 removal technologies are on the order of 100-300) kg. In addition to their impact on the AR subsystem, even the WM subsystem weights are seen to be affected somewhat by the choice of CO_2 removal technology. Such interactions between different subsystems cannot be recognized quantitatively by comparing the weight, power demand, etc. of individual technologies by themselves. For example, the solid amine water resorption (SAWD) process puts steam into the cabin air, which is condensed and the condensate becomes an additional load on the hygiene water processing unit, thereby increasing its weight and power demand. Because of the increased throughput, any nonregenerable chemicals used by hygiene water processing also increases and can be accounted for in the increased storage subsystem weight. The LiOH CO_2 removal technology is for nonregenerative capture of CO_2. The weight of the LiOH sorption equipment itself is small compared to the other regenerative CO_2 removal. process units. However, since the process is nonregenerable, there is a high demand for LiOH canisters (as seen in the storage subsystem weight), which is directly proportional to crew size and mission duration.

Subsystem power demands also show significant differences. The power demand for the various CO_2 removal technologies is less by hundreds of watts compared to the baseline four-bed molecular sieve (4BMS) with the exception of the air polarized concentrator (APC) . Even though the electrochemical depolarized concentrator (EDC) shows a marked decrease in power demand for the CO_2 removal SFE, the power advantage does not carry through exactly into the AR subsystem. EDC adversely affects the AR subsystem by requiring additional H_2 generation and thus increasing the size, throughput, and power demand on the water electrolysis unit. LiOH requires the minimum power for the SFE, AR subsystem, and the overall system since the LiOH technology has low power and the CO_2 reduction process is eliminated.

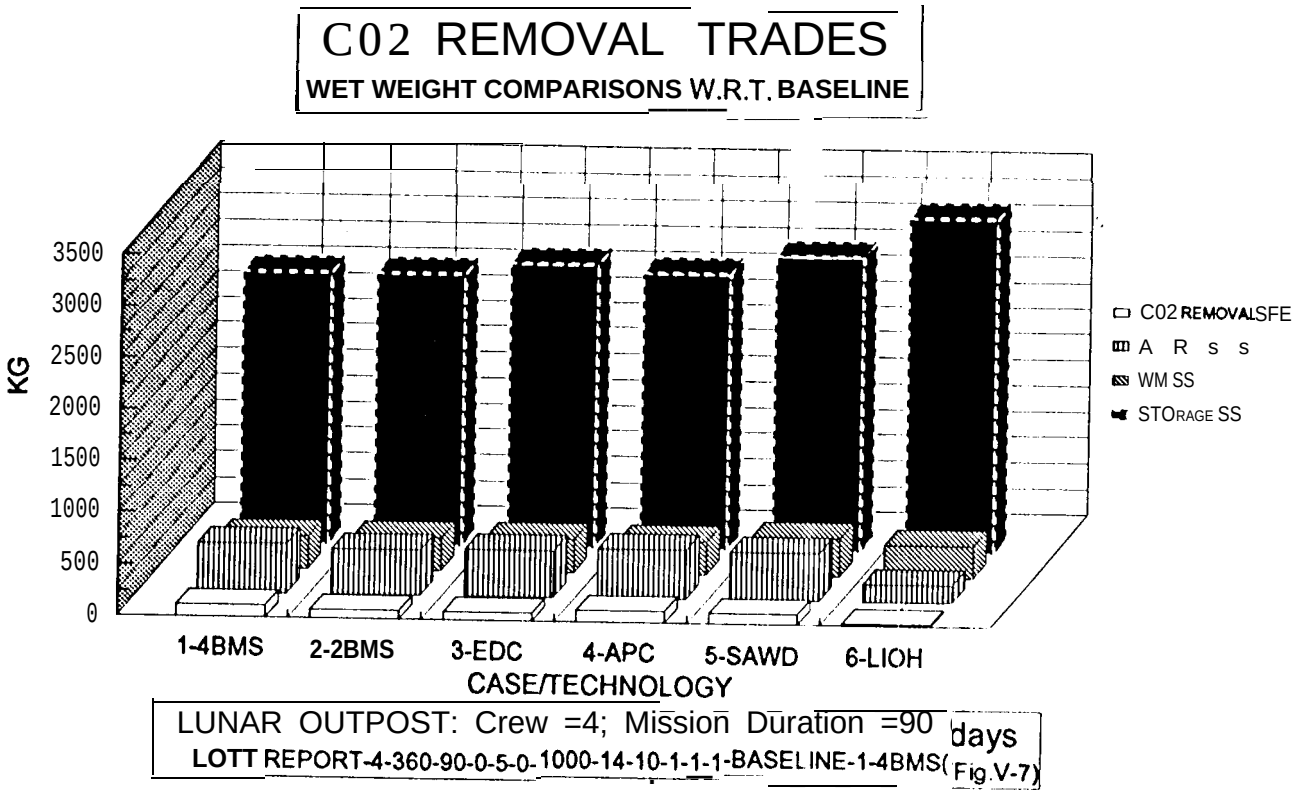


Figure v-7 . CO_2 Removal Trade Weight Comparisons

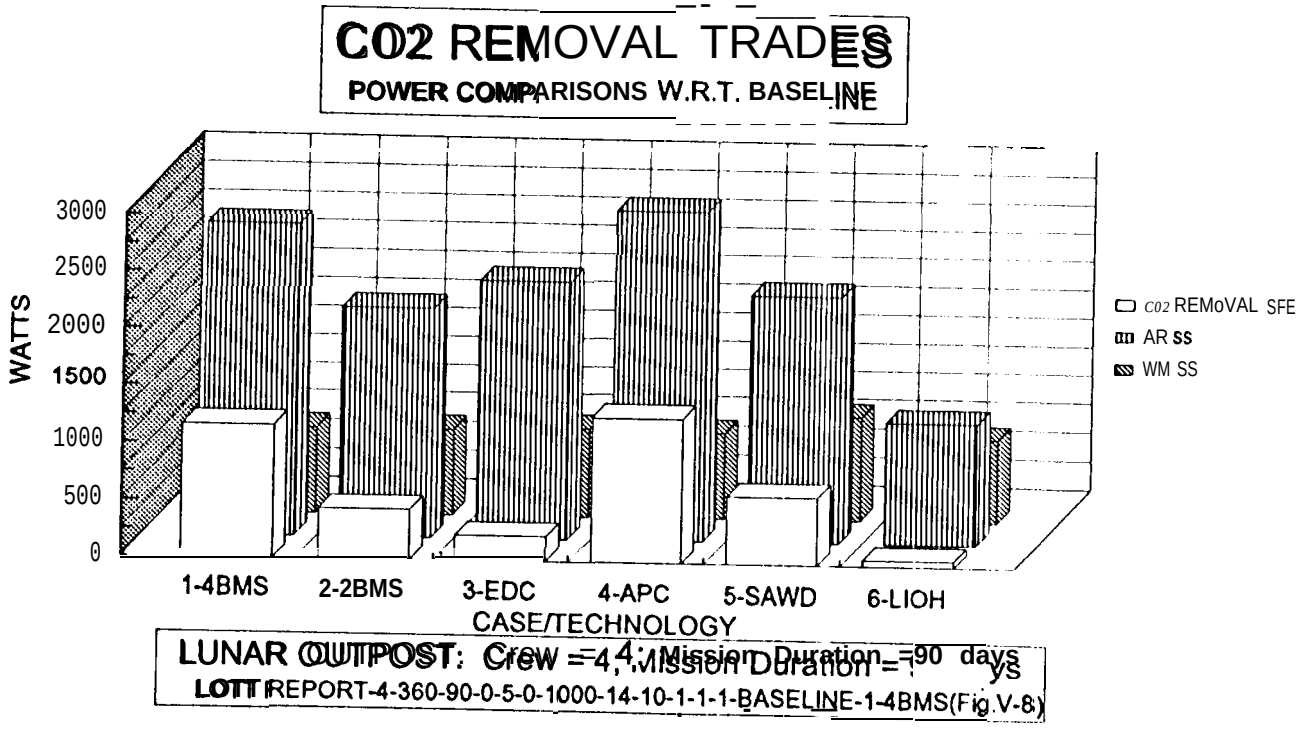


Figure V-8. CO_2 Removal Trade Power Comparisons

3. CO_2 Reduction Technology Trade

The baseline uses Bosch technology to recover O_2 as water condensate and is compared for subsystem weights and power demands to the Sabatier advanced carbon removal system (ACRS), and the CO_2 electrolysis/Boudouard (CO2EL/BD or CO2EL) 'process in Figures V-9 and V-10, respectively. The technology choice here has no intersubsystem impact except for the storage subsystem. The simplest of the four processes in terms of weight and power is the Sabatier process, which catalytically converts all of the $C0_2$ in its feed to CH_4 by reacting with H_2. However, the H_2 requirement places an additional burden on the O_2 generation SFE of the AR subsystem, thereby losing its advantage over other technologies. Since the CH_4 produced by Sabatier technology is vented as trash, the associated H_2 loss must be supplied by additional storage of hydrogen or preferably water, which is reflected in the higher storage subsystem weight.. For the baseline system, using Bosch, there is a net requirement of 0.8 kg per day of makeup water for a crew of 4; with the Sabatier process, this makeup water increases to 3.7 kg per day. However, the Bosch process also requires chemical supplies in the form of canisters to collect the carbon formed in the process. These canisters account for 0.5 kg per day. Hence, the net consumables difference per day between the Sabatier and the Bosch processes is 2.4 kg, which amounts to over 200 kg for a 90-day mission. Another way of configuring the system with the Sabatier process would be to convert only part of the CO_2 produced. This scheme would take only available H_2 created from the 0_2 generation SFE due to metabolic O_2 requirements. This would reduce the size of the O_2 generation unit significantly as the stoichiometric ratio of H_2/CO_2 requirement for Bosch is 2 and for Sabatier is 4 for complete CO_2 reduction. The impact would significantly affect the power requirements for the $C0_2$ reduction and H_2O electrolysis processes.

The ACRS and C02EL processes show results comparable to the baseline Bosch process in terms of weight; ACRS shows slightly higher power than Bosch for both the SFE and AR subsystem, while C02EL shows a higher SFE power but a slightly lower AR subsystem power.

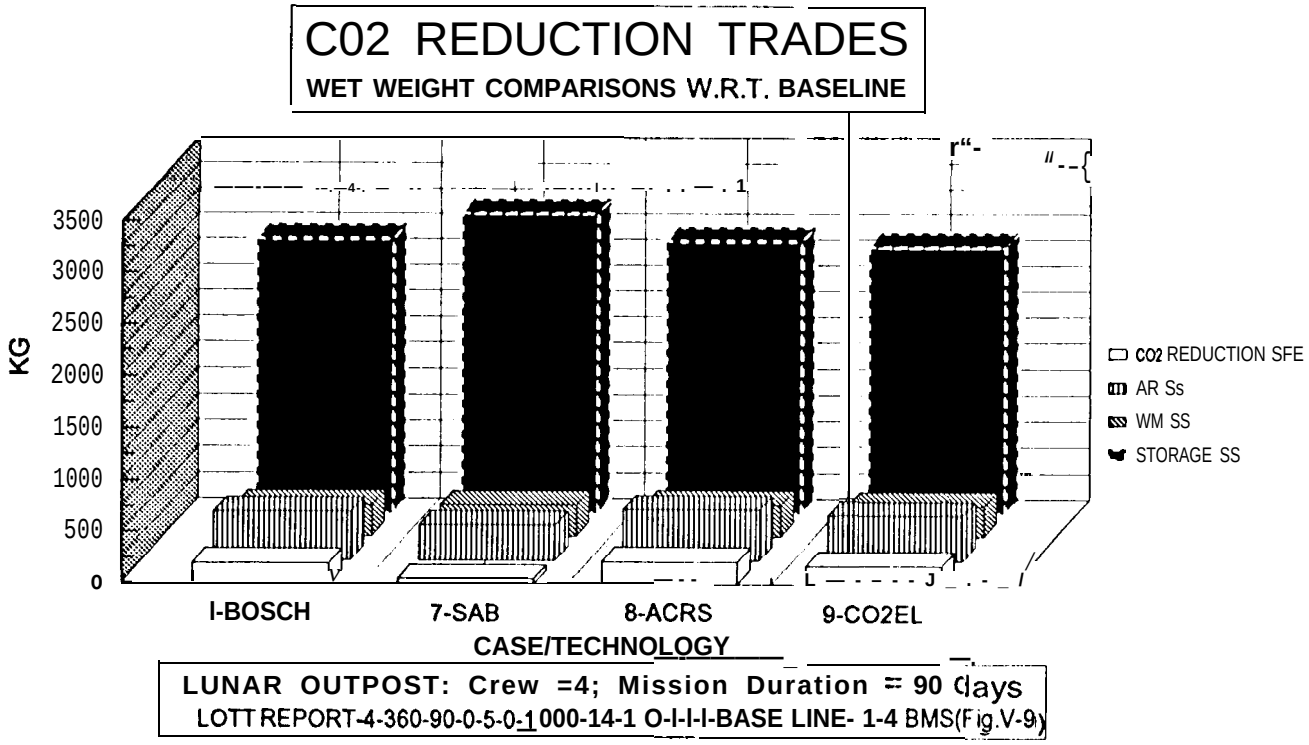


Figure v-9 . CO_2 Reduction Trade Weight Comparisons

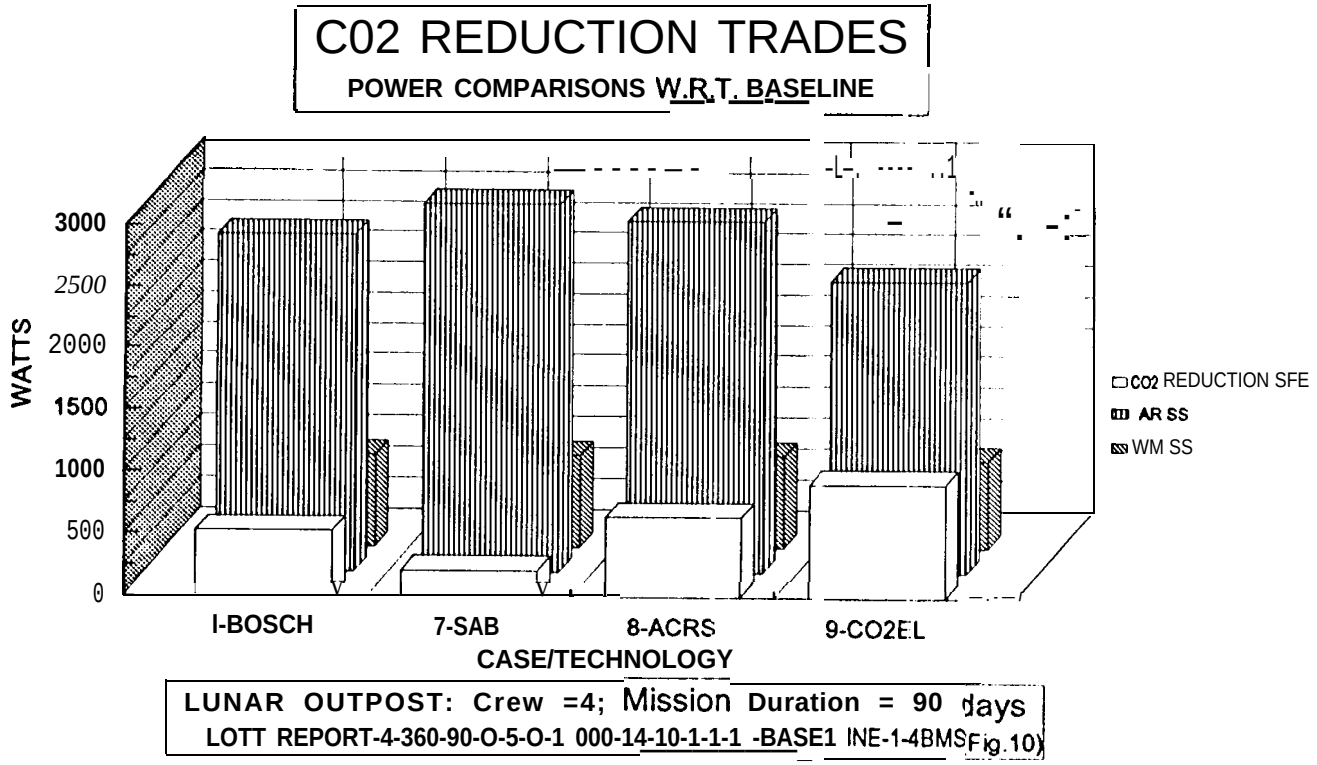


Figure V-10 . CO_2 Reduction Trade Power Comparisons

4. O_2 Generation Technology Trade

The O_2 generation subsystem functional element uses the static feed water electrolysis (SFWE) process as its baseline. SFWE is compared to the subsystem weight and power parameters for water vapor electrolysis (WVE) and solid polymer electrolyte liquid feed (SPE) in Figures V-II and V-12.

SFWE and SPE compare closely both in weight and power demand, with SFWE having only a slight advantage due to the lower weight and power demands at the SFE level. However, the WVE affects both the WM and storage subsystems because the WVE process draws water out of the cabin air and then electrolytes the H_2O to H_2 and O_2. This avoids the condensation of atmospheric moisture and the subsequent cleaning of condensate water to standards of purity required for electrolysis. The net effect is to reduce the magnitude of condensate processing imposed on the WM subsystem and thereby reducing the WM subsystem weight., power, and chemical supplies by that required for condensate treatment. This then results in the lowest overall system weight as shown in Figures V-1 (90 days) and V-2 (600 days).

The comparison of power demand numbers shows that WVE results in significantly lower overall system power by over 200 watts. The primary reduction is seen at the O_2 generation SFE level. A slight reduction is also realized in the WM subsystem.

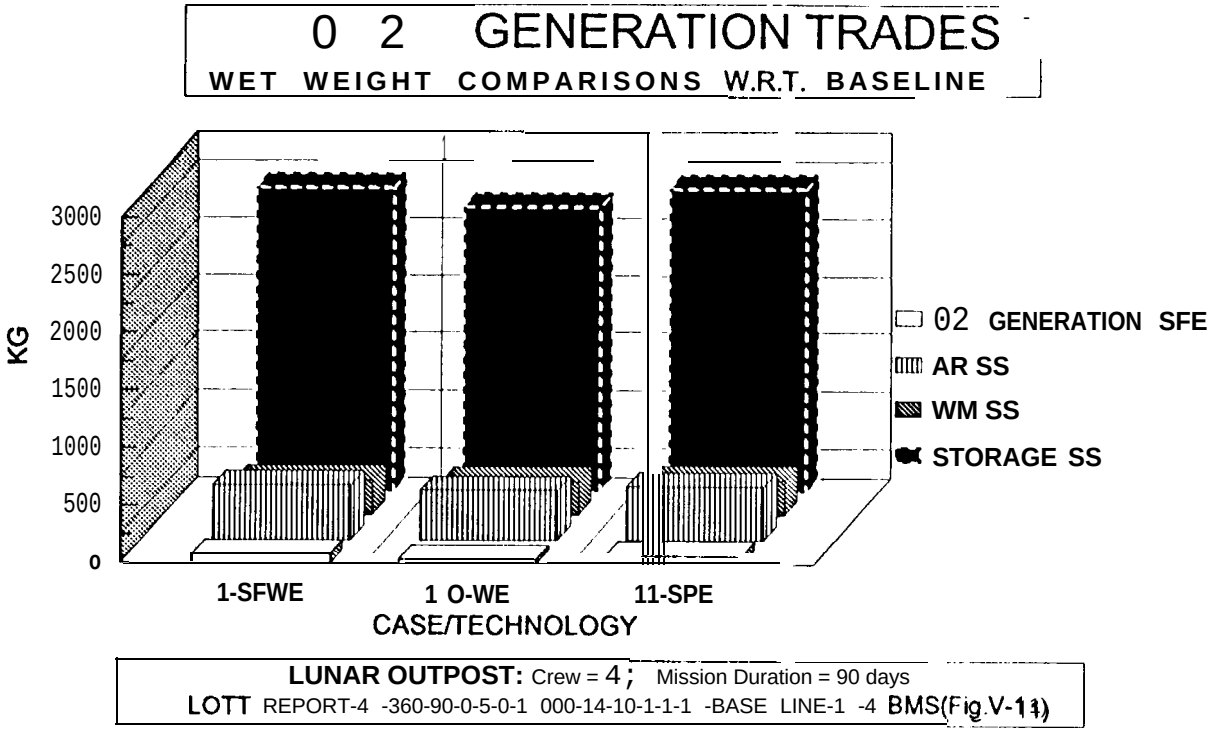


Figure V-II . O_2 Generation Trade Weight Comparisons

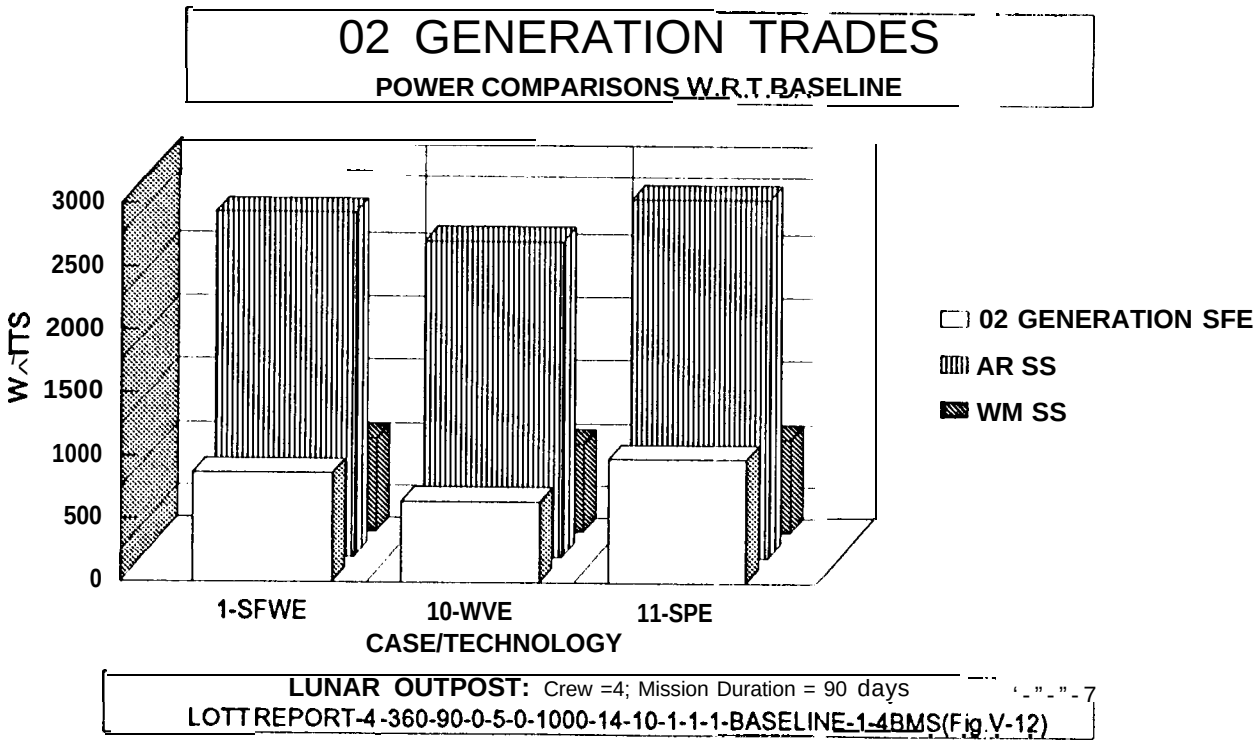


Figure v-12. O_2 Generation Trade Power Comparisons

5. Potable H_2O Processing Technology Trade

The subsystem functional element for potable water recovery uses multifiltration for potable water (MFPW or ME') as its baseline which is compared to the subsystem weights and power of reverse osmosis for potable water (ROPW or RO) and electrochemical deionization (ELDI) in Figures V-13 and V-14.

RO and ELDI recover less water (~90%) compared to the baseline value of 99.99%, thereby showing a higher storage subsystem weight to carry the extra makeup water not recovered; this represents about a 2 kg per day difference in water. However, the higher water recovery rate for MF is tempered by a higher demand for consumable chemicals (MF unibeds) compared to the RO. The weights computed for the potable water recovery SFE and for the WM and AR subsystem are similar for all the three processes; the storage subsystem is lowest for the MF as it recovers the most water.

Power demand for the MF and RO is essentially equal, while ELDI shows a significantly higher rate. Other SFES and subsystems are not affected by the change in the technology candidate for potable H_2O processing. On the other hand, if it would be possible to route the RO brine from potable water processing to urine processing, then the overall water recovery could be increased at the expense of higher SFE weight and power demand of urine processing. It would also be possible to compute the mission duration for a break-even point where the reduced water supply requirement matches the increased weight and power demand (equating incremental power to weight) for urine processing.

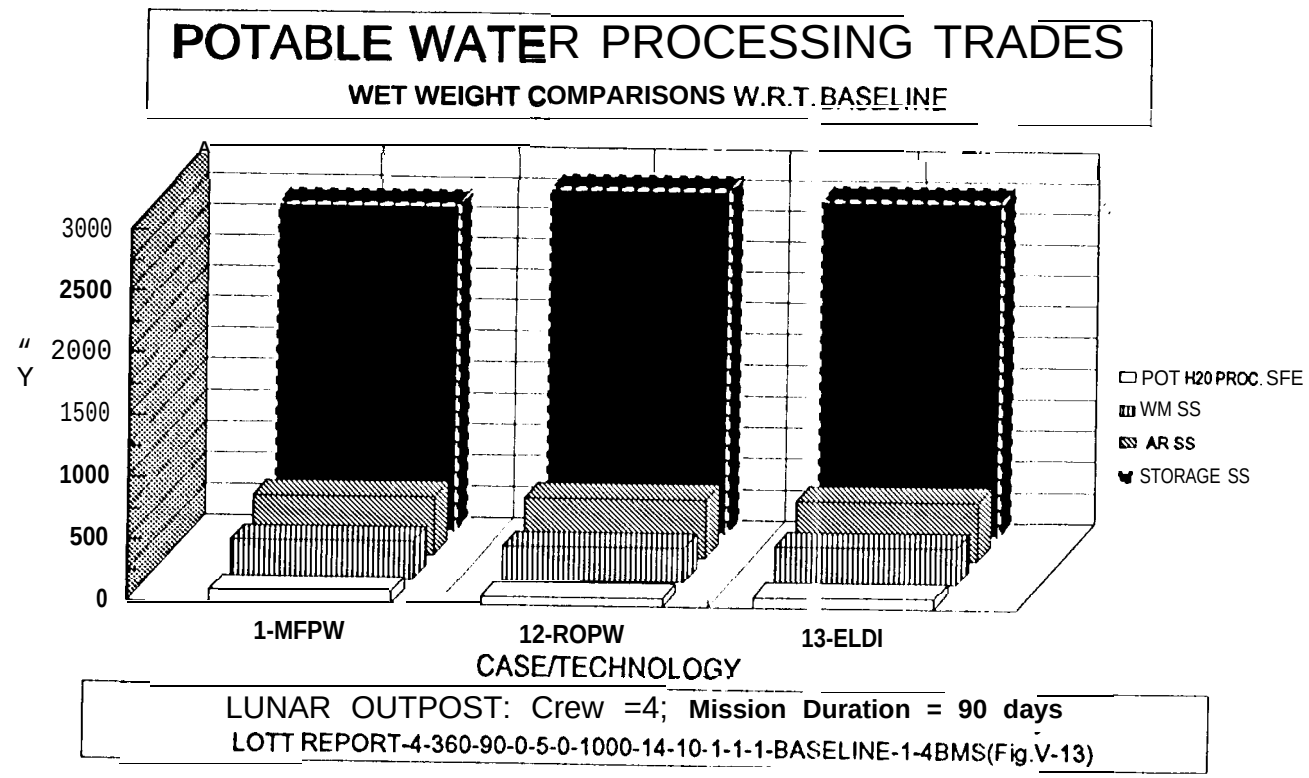


Figure V-13 . Potable Water Processing Trade Weight Comparisons

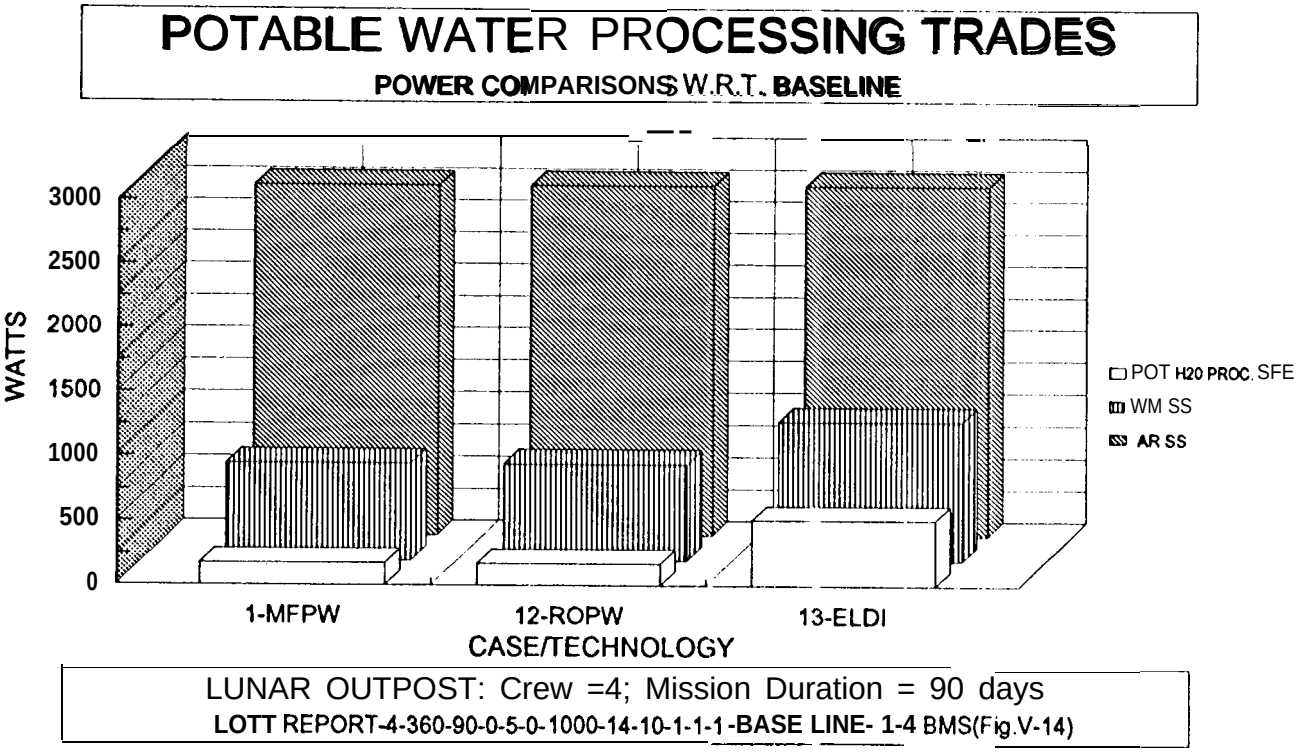


Figure V-14. Potable Water Processing Trade Power Comparisons

6. Hygiene H_2O Processing Technology Trade

The reverse osmosis for hygiene water (ROHW or RO) baseline process has a lower water recovery rate (93.5%) compared to the 99.99% recovery for multifiltration for hygiene water (MFHW or MF). By switching to MF, the system completely regenerates all the hygiene water requirement: in fact, an excess of H_2O is generated, which must be stored as trash or dumped overboard. For the baseline ROHW process, the makeup rate for four persons is 0.8 kg per day and for the MFHW process, there is no demand for makeup. However, in treating all of the wash waters, the MF process consumes an additional 1.1 kg per day of ion exchange and adsorption beds (unibeds), thereby causing a net increase in consumable supplies of 0.3 kg per day compared to the RO process. The overall impact on the storage subsystem is small (less than 50 kg). The primary weight difference between the two cases is mostly attributed to the weights of the RO and MF processes with the ROHW weighing about 100 kg more than the MFHW.

The power demands for RO and MF *are* compared in Figure V-16. The MF shows a power decrease relative to the RO process of over 300 watts at the SFE level. This difference accounts for the entire difference at the system level; i.e., the choice between RO and MF limits their comparison at the SFE level since neither of them have any impact on other SFES or subsystems with respect to power demand.

An option for RO would be to route the RO brine to urine processing thereby increasing the overall H_2O recovery depending on the recovery rate of the urine processing technology selected.

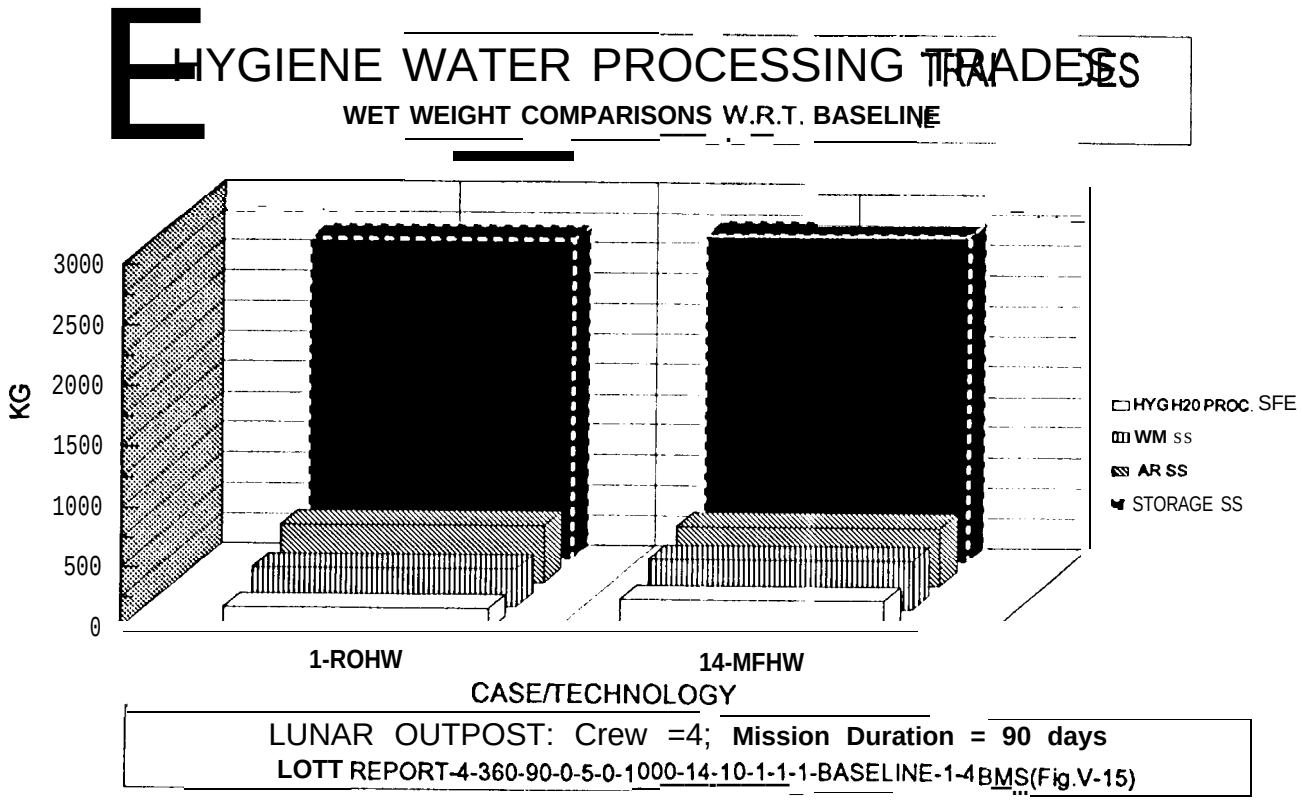


Figure v-15. Hygiene Water Processing Trade Weight Comparisons

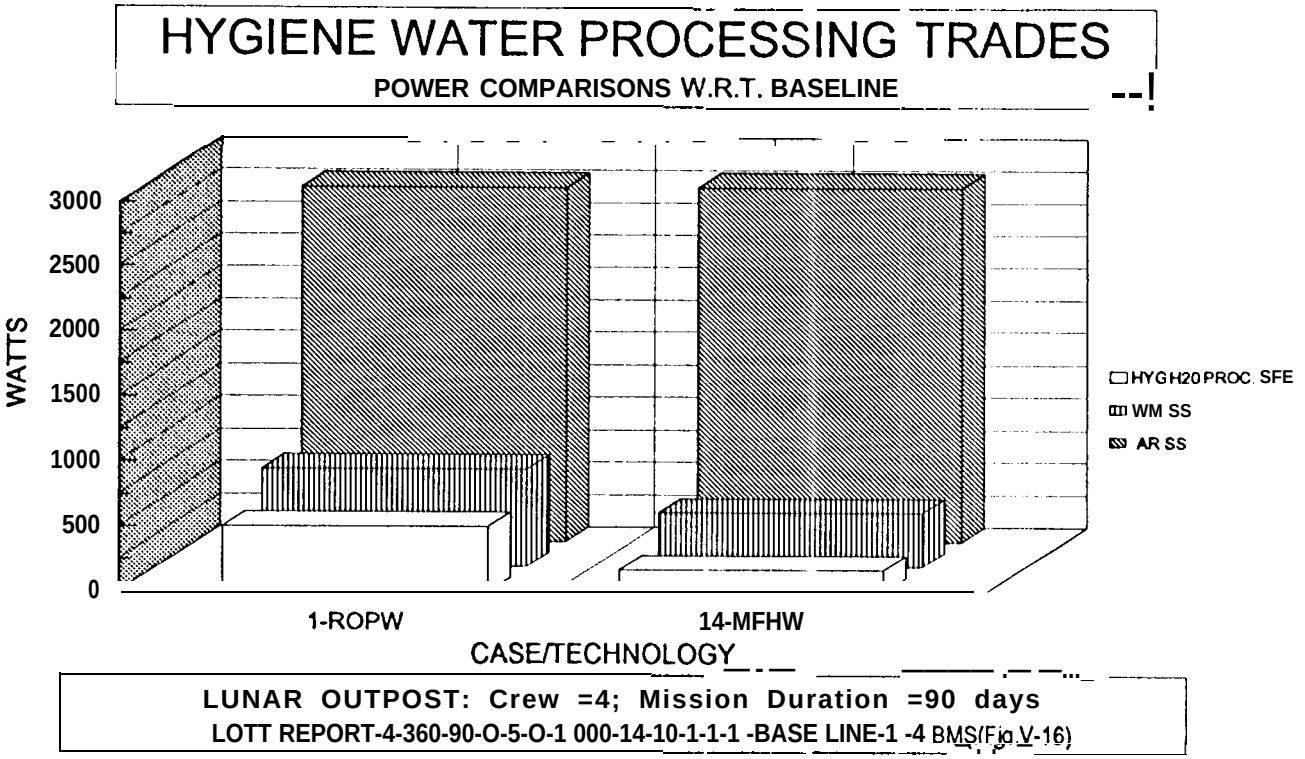


Figure V-16 . Hygiene Water Processing Trade Power Comparisons

7. Urine Processing Technology Trade

Thermoelectric integrated membrane evaporation system (TIMES) technology, as the baseline for urine processing, was compared in terms of the impact of substitution with vapor compression distillation (VCD), vapor phase catalytic ammonia removal (VPCAR), and air evaporator (AIRE) processes in Figures V-17 and V-18.

Water recovery rates for the TIMES baseline, VCD, VPCAR, and AIRE range from 90% for VCD and VPCAR to 99.9% for the AIRE process, respectively, resulting in small differences in storage subsystem weights relating to makeup water requirement. Makeup water for the TIMES baseline is 0.8 kg per day for a crew of 4; for VCD, VPCAR, and AIRE, the makeup rates are 1 .5, 1.4, and 0.7 kg per day, respectively. While the AIRE has the highest water recovery, there is a significant weight associated with the use of wicks as a nonregenerable chemical supply that amounts to 0.6 kg per day. The overall weight effect is that the TIMES and AIRE cases are similar and the VCD and VPCAR are slightly higher due to lower water recoveries.

Power demand shows the AIRE and the VPCAR processes requiring about 100 watts more than the TIMES and the VCD for the urine processing SFE. VPCAR also requires slightly more power from the AR subsystem, as it requires additional oxygen generation for NH_3 oxidation.

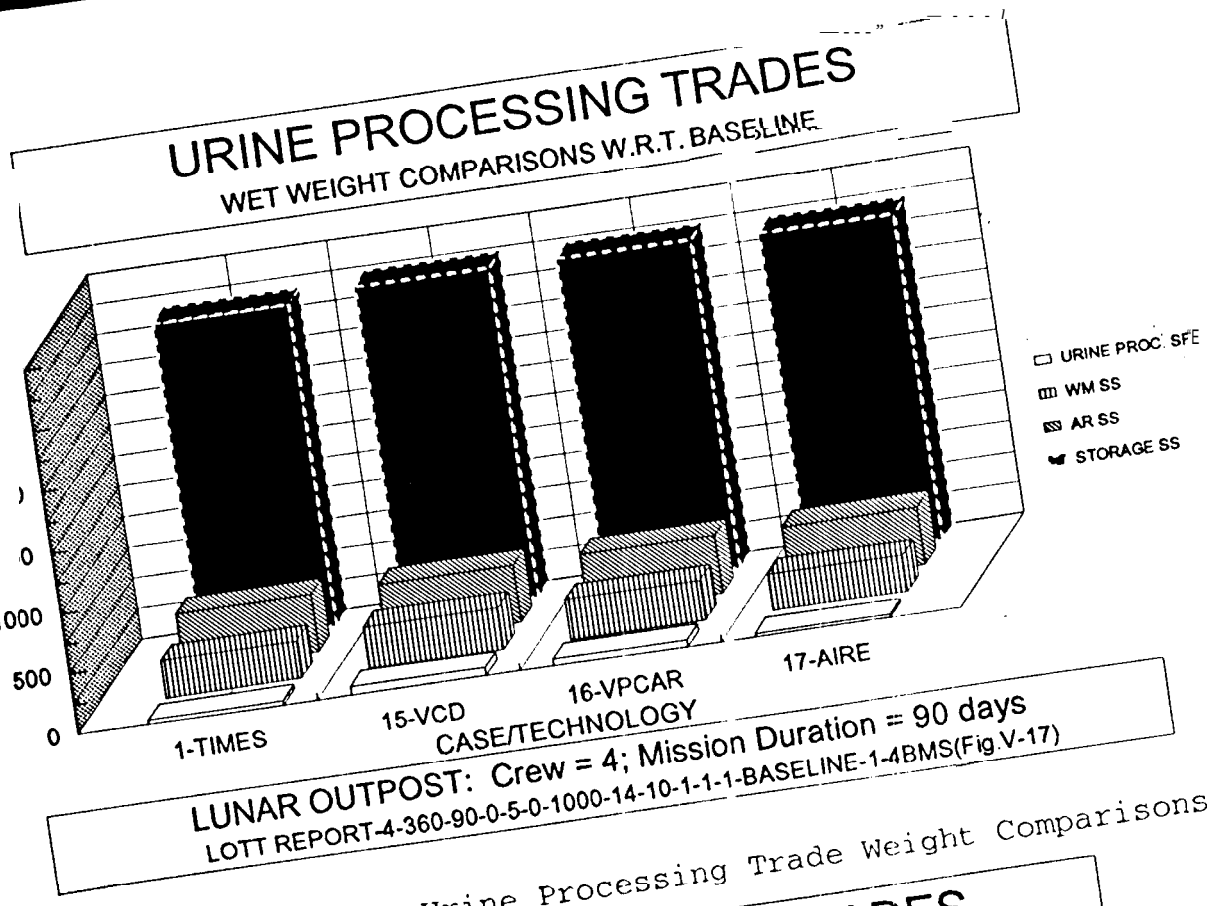


Figure V-17. Urine Processing Trade Weight Comparisons

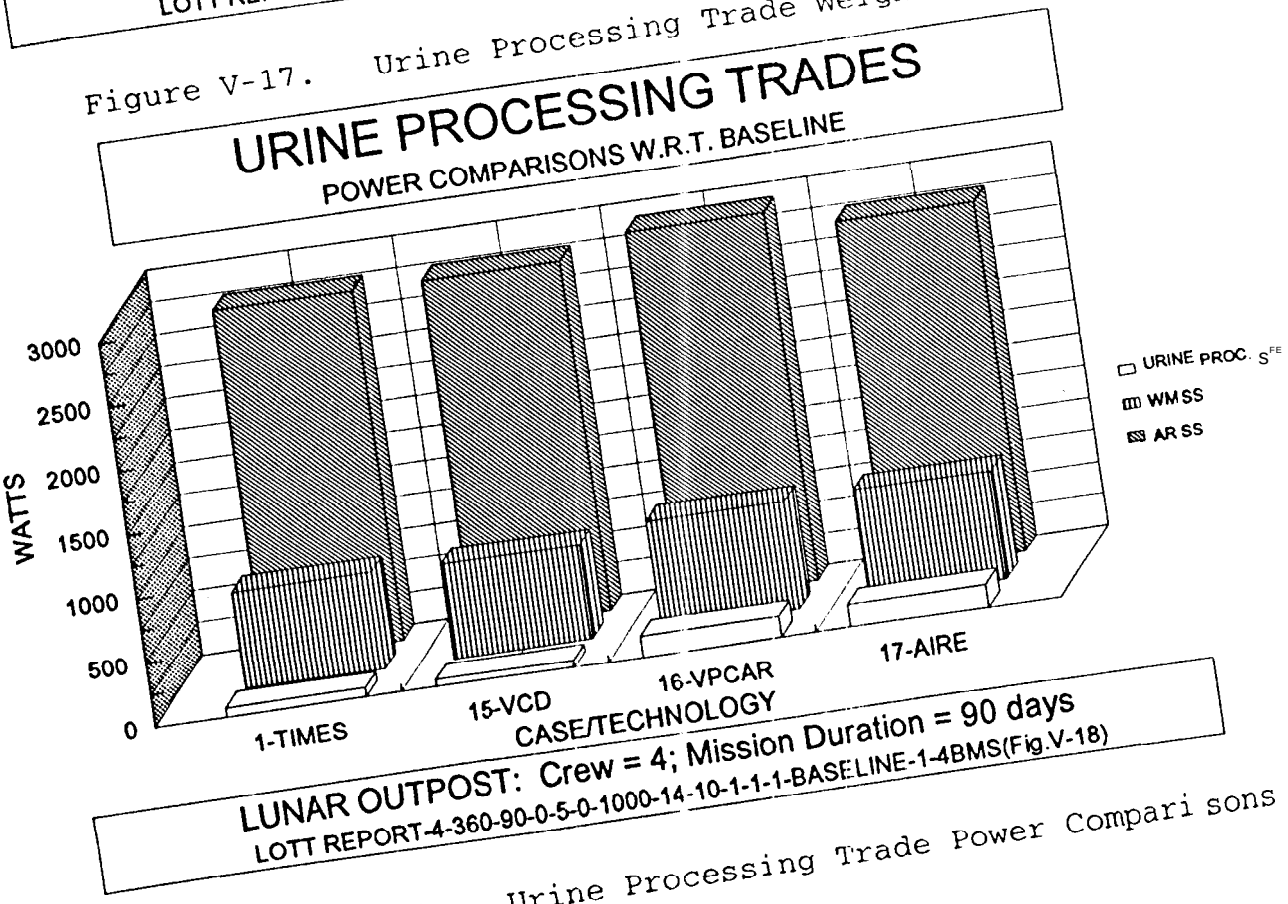


Figure V-18. Urine Processing Trade Power Compari sons

8. Solid Waste Treatment Technology Trade

The baseline system does not use solids waste treatment. In Figures V-19 and V-20, it is compared to subsystem weight and power demand for freeze drying (FD), thermal drying (TD), combustion (COMB), wet oxidation (WOX), and super critical water oxidation (SCWO).

System weight increases over the baseline are 260, 60, 90, 170, and 280 kg for FD, TD, COMB, WOX, and SCWO, respectively. For the FD and TD processes, the weight increases are mostly attributed to the weight of the drying equipment, as shown in Figure V-19. The drying processes produce water condensate that must be treated in the WM subsystem. In the case of the oxidation processes, additional condensate is produced by the oxidation of organic solids. The CO_2 and trace pollutant gases released by oxidation are considered to be a concentrated polluted gas stream that must be treated by trace contaminant control in the AR subsystem for pollutant oxidation, carbon rejection, and oxygen recovery. Hence, the oxidation processes affect both the WM and AR subsystems, while the drying processes impact the WM subsystem only.

For the SCWO process, it has been reasonably assumed that the condensate produced from the process can be routed directly to the potable water bus where it could be mixed with other water produced from the WM subsystem such that an acceptable average water quality for potable water is achieved.

Storage subsystem weights are slightly higher (about 40 kg) than the baseline for the drying processes. The weight savings in makeup water is offset by the additional chemical supplies required for the SWT and WM subsystems. For the oxidation processes, the storage subsystem weights are higher by 110 kg for COMB and WOX and by only 50 kg for the SCWO process. Similar to the drying processes, the savings in makeup water weight is offset by the additional chemical supplies for SWT and WM subsystems; for SCWO, no additional chemical supplies for WM is required as its condensate is sent directly to the potable water bus without having it processed in the WM subsystem. However, since the oxidation process requires additional gas processing, storage weights of several waste gases (such as O_2, CO_2, H_2, and Concentrated Polluted Gas Mix) are slightly increased.

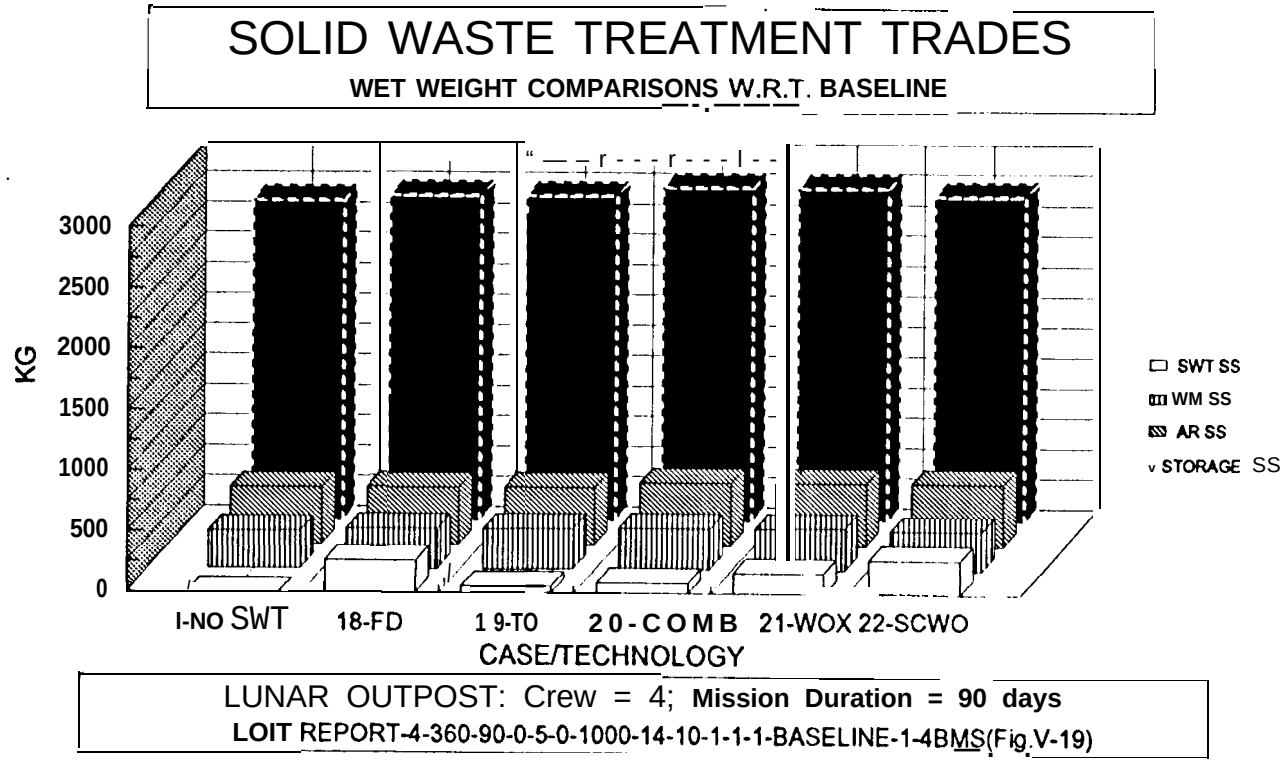


Figure V-19 . Solid Waste Treatment Trade Weight Comparisons

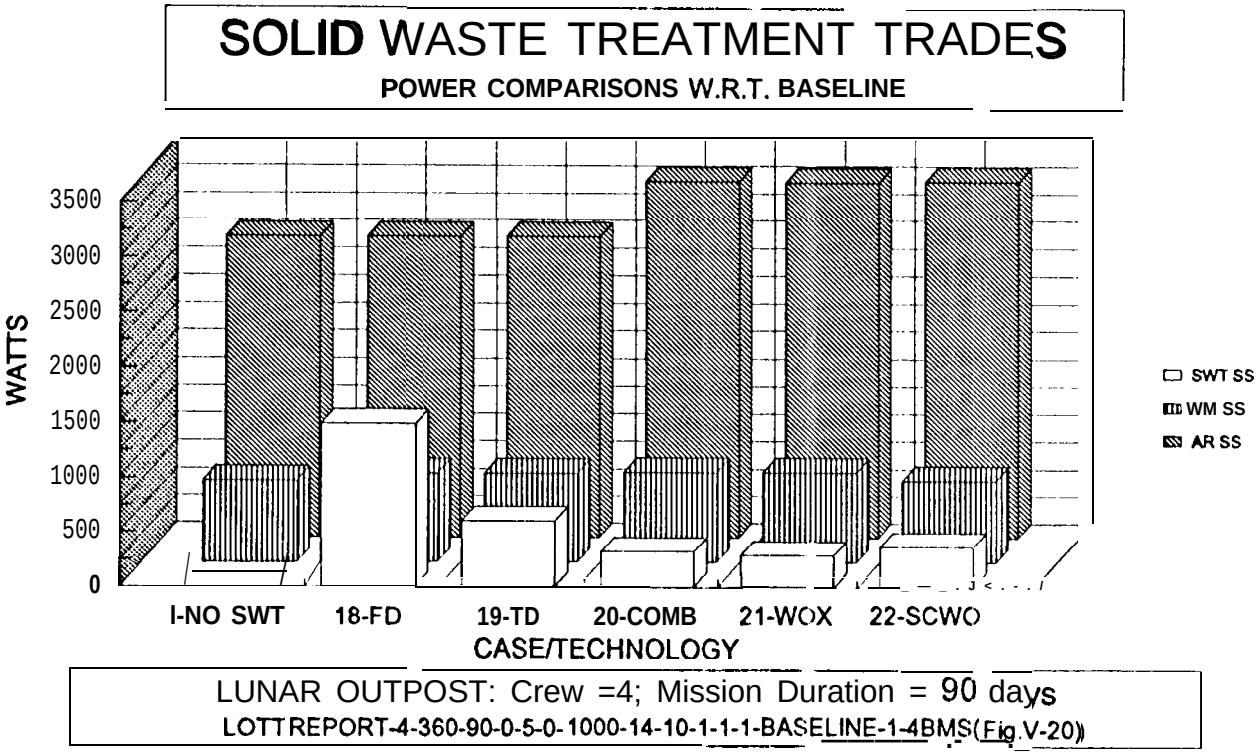


Figure V-20 . Solid Waste Treatment Trade Power Comparisons

Increases in power demand over the baseline are 1580, 700, 950, 910, and 920 watts for FD, TD, COMB, WOX, and SCWO, respectively. The additional power demands attributed to the SWT subsystem alone are 1490, 600, 330, 290, and 370 watts, respectively. For the drying processes, the power increases are predominantly due to the drying processes themselves with a slight contribution from the WM subsystem for processing of additional water condensate. For the oxidation processes, more than half of the power increase can be attributed to the additional gas processing required of the AR subsystem; the power demand for oxidation contributes slightly less than half of the additional power required. For COMB and WOX, there is a slight additional power demand on the WM subsystem similar to the drying processes; for SCWO, there is no additional load placed on the WM subsystem as its product condensate goes directly to the potable water bus.

Surplus Water and Food-Water:

Feed to the SWT subsystem includes feces from the human habitat and brines from the WM subsystem. All. SWT cases provide for additional water recovery leading to a surplus of water developed which must be trashed. For the drying processes (FD and TD), the surplus amounts to 5.7 kg per day for a crew size of 4. Table V-2 illustrates a metabolic balance for a crew size of 4. Note that the ratio of food-water content to dry-food constituents is 1.83. With this quantity of water contained in the stored food, there is an excess of water produced as a result of using solids waste processing. If it is assumed that stored food can be reduced significantly to levels such as freeze-dried food, then the weight of stored food and the amount of excess water produced will decrease accordingly.

Table V-2. Metabolic Balance for Crew of 4:
(1.83 food-water-to-dry-food ratio)

INPUTS:	CARBON (kg)	HYDROGEN (kg)	OXYGEN (kg)	NITROGEN (kg)	ASH (kg)	TOTALS (kg)
1. DRY FOOD						
Protein, C4H50N	0.3080	0.0324	0.1028	0.0896		0.5328
Carbohydrate, C6HI 206	0.5956	O.loot	0.7936			1.4892
Fat, C16H32O2	0.3432	0.0576	0.0572			0.4580
Minerals, Ash					0.0380	0.0380
2. LIQUIDS (WATER)						
Drink		0.7208	5.7192			6.4400
Food Preparation		0.3536	2.8064			3.1600
Food Water Content (1 .83*dry food)		0.5148	4.0852			4.6000
3. GASES						
Oxygen			3.3436			3.3438
INPUTSUMS	1.2468	1.7792	16.9080	0.0896	0.0380	20.0616
OUTPUTS:						
1. SOLID WASTES						
Unne,C2H602N2	0.0640	0.0160	0.0852	0.0748	0.0308	0.2708
Feces, C42H69O13N5	0.0708	0.0096	0.0292	0.0096	0.0072	0.1264
Sweat,C13H28013N2	0.0296	0.0056	0.0396	0.0052		0.0800
2. LIQUIDS (WATER)						
Urine		0.6776	5.3764			6.0540
Feces		0.0408	0.3224			0.3632
Sweat & Perspiration		1.0296	8.1716			9.2012
3. GASES						
Carbon dioxide	1.0824		2.8836			3.9660
OUTPUT SUMS	1.2468	1.7792	16.9080	0.0896	0.0380	20.0616
Potable water recycled:		1.0744	8.5256			**9.6000**
Potable water with stored food:		0.5148	4.0852			**4.6000**
Total water in:		1.5892	12.6108			**14.2000**
Total water out:		1.7480	13.8704			**15.6184**
Net water metabolized:		0.1588	1.2596			**1.4184**
= = + -						
Excess water produced (HDfi1 " T 6 4 1 8			5.0940			5.7358

Lowering the Food Water Content:

Table v-3 shows the same metabolic balance using a ratio of 0.01 food-water-to-dry-food (0.01 is used here for illustration purposes and is not meant as a suggested food composition). In both tables, the crew is ingesting the same water and food and producing the same outputs. Decreasing the food water content requires an increase in the recycled potable water from 9.6 to 14.2 kg per day while decreasing the excess water produced from 5.7 kg to 0.5 kg per day if thermal drying is used for solid waste treatment. Note that 1.4 kg of water are created metabolically regardless of the food water content.

For the oxidation processes, the surplus is 6.2 kg per person day for the higher food water content. Creating this surplus comes at the expense of weight and power. The oxidation processes effectively create more water by oxidizing the solids waste to CO_2 and H_2O.

Table V-3. Metabolic Balance for Crew of 4:
(0.01 food-water-to-dry-food ratio)

INPUTS:	CARBON (kg)	HYDROGEN (kg)	OXYGEN (kg)	NITROGEN (kg)	ASH (kg)	TOTALS (kg)
1. DRY FOOD						
Protein, C_4H_5ON	0.3080	0.0324	0.1028	0.0896		0.5328
Carbohydrate, $C_6H_{12}O_6$	0.5956	0.1000	0.7936			1.4892
Fat, $C_{16}H_{32}O_2$	0.3432	0.0576	0.0572			0,4580
Minerals, Ash					0.0380	0.0380
2. LIQUIDS (WATER)						
Drink		0.7208	5.7192			6.4400
Food Preparation		0.8656	6.8692			7.7348
Food Water Content (0.01 .dry food)		0.0028	0.0224			0.0252
3. GASES						
Oxygen			3.3436			3.3436
INPUT SUMS	1.2468	1.7792	16.9080	0.0896	0.0380	20.0616
OUTPUTS:						
1. SOLID WASTES						
Urine, $C_2H_6O_2N_2$	0.0640	0.0160	0.0852	0.0748	0.0308	0.2708
Feces, $C_{42}H_{69}O_{13}N_5$	0.0708	0.0096	0.0292	0.0096	0.0072	0.1264
Sweat, $C_{13}H_{28}O_{13}N_2$	0.0296	0.0056	0.0396	0.0052		0.0800
2. LIQUIDS (WATER)						
Urine		0.6776	5.3764			6.0540
Feces		0.0408	0.3224			0.3632
Sweat & Perspiration		1.0296	8.1716			9.2012
3. GASES						
Carbon dioxide	1.0824		2.8836			3.9660
OUTPUT SUMS	1.2468	1.7792	16.9080	0.0896	0.0380	20.0616
Potable water recycled:		1.5864	12.5884			14.1748
Potable water with stored food:		0.0028	0.0224			0.0252
Total water in:		1.5892	12.6108			14.2000
Total water out:			13.8704			15.6184
Net water metabolized:		0.1588	1.2596			1.4184
Excess water produced (HD):		0.0507	0.4039			0.4546

9. Equivalent System Penalty Weight Comparisons

By assigning a weight value to the incremental power required for different life support technologies, an equivalent system weight can be calculated and compared to the baseline technology used. For this report, a regenerative fuel cell technology [reference V-1] has been assumed using a value of 3 watts/kg for the incremental power. The life support system weight is added to the equivalent power weight to represent a total equivalent life support weight. In this manner, penalties relative to the baseline system weights are compared for air revitalization, water management, and solid waste treatment technologies. Penalties therefore represent additional mass that must be lifted to the lunar surface relative to baseline technologies used in Case 1.

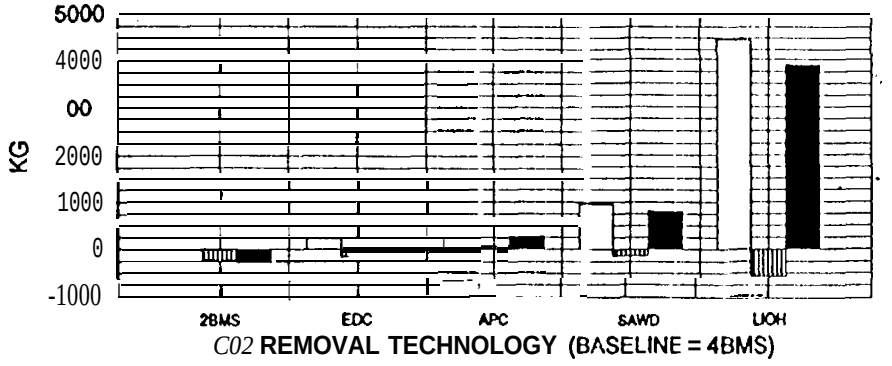
Air Revitalization Technologies:

Figure V-21 shows all of the AR technologies for the 4-person/600-day mission. For the CO_2 removal processes, the largest penalty relative to the baseline 4BMS is associated with LiOH. The 2BMS shows an advantage (negative penalty value) of 280 kg; most of these advantages are attributed to power.

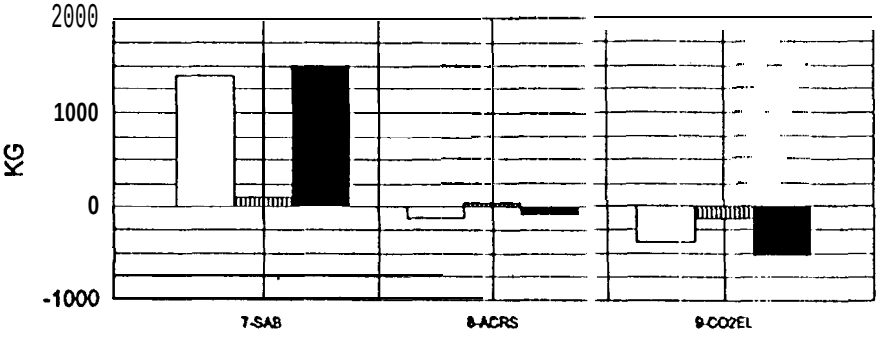
For the CO_2 reduction processes, SAB shows a significant weight penalty relative to the Bosch baseline, while ACRS and C02EL show total. equivalent advantages of 80 and 500 kg respectively. For the O_2 generation technologies, the WVE shows a significant total equivalent advantage of 600 kg relative to the SFWE baseline. This advantage is mostly attributable to lower storage supplies for water processing of condensate; the WVE process effectively removes moisture from the air, thereby reducing the amount of condensate to be treated in potable water processing. SPE is essentially identical to SFWE.

SYSTEM WEIGHT PENALTY COMPARISONS

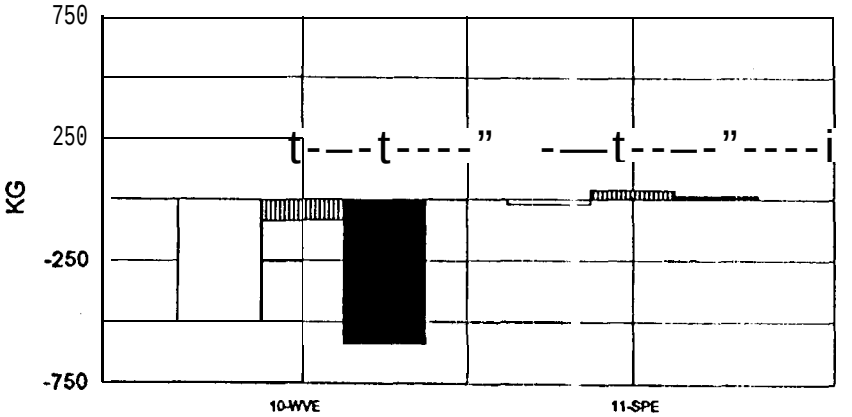
EQUIVALENT WEIGHT PENALTY W.R.T. BASELINE AR TECHNOLOGIES



C02 **REMOVAL TECHNOLOGY (BASELINE = 4BMS)**



C02 REDUCTION TECHNOLOGY (BASELINE = BOSCH)



02 GENERATION TECHNOLOGY (BASELINE = SFWE)

☐ System penalty ▨ Equivalent power penalty@ 3 watts/kg

■ **Total equivalent system penalty**

LUNAR OUTPOST: crew= 4; Mission Duration = 600 days

LOTT REPORT< -2400-600-0-5-0-1000-14-10-1-1-1-BASELINE-1-4BMS(Fig.V-21)

Figure V-21. Equivalent System Weight Comparisons : Air Revitalization Technologies

Water Management Technologies:

Figure V-22 shows the total equivalent system penalties for all of the WM technologies. For the potable water processes, the weight disadvantages are due to lower water recoveries; since brines are not processed in this configuration, unrecovered water must be made up from storage.

For the hygiene water processes, there is a penalty of using MF relative to RO for using additional unibed material (which shows up as a consumable item in the storage subsystem) . However, there is a power advantage of the MF system that roughly decreases the disadvantage of storage supplies by one-third.

For the urine processing technologies, equivalent power weights are similar to the baseline at the system level. Differences in system level penalties for VCD anti VPCAR are attributed to water recoveries. For the AIRE process, even though the water recovery is nearly 100%, there is a penalty associated with expendable wicks amounting to over 200 kg.

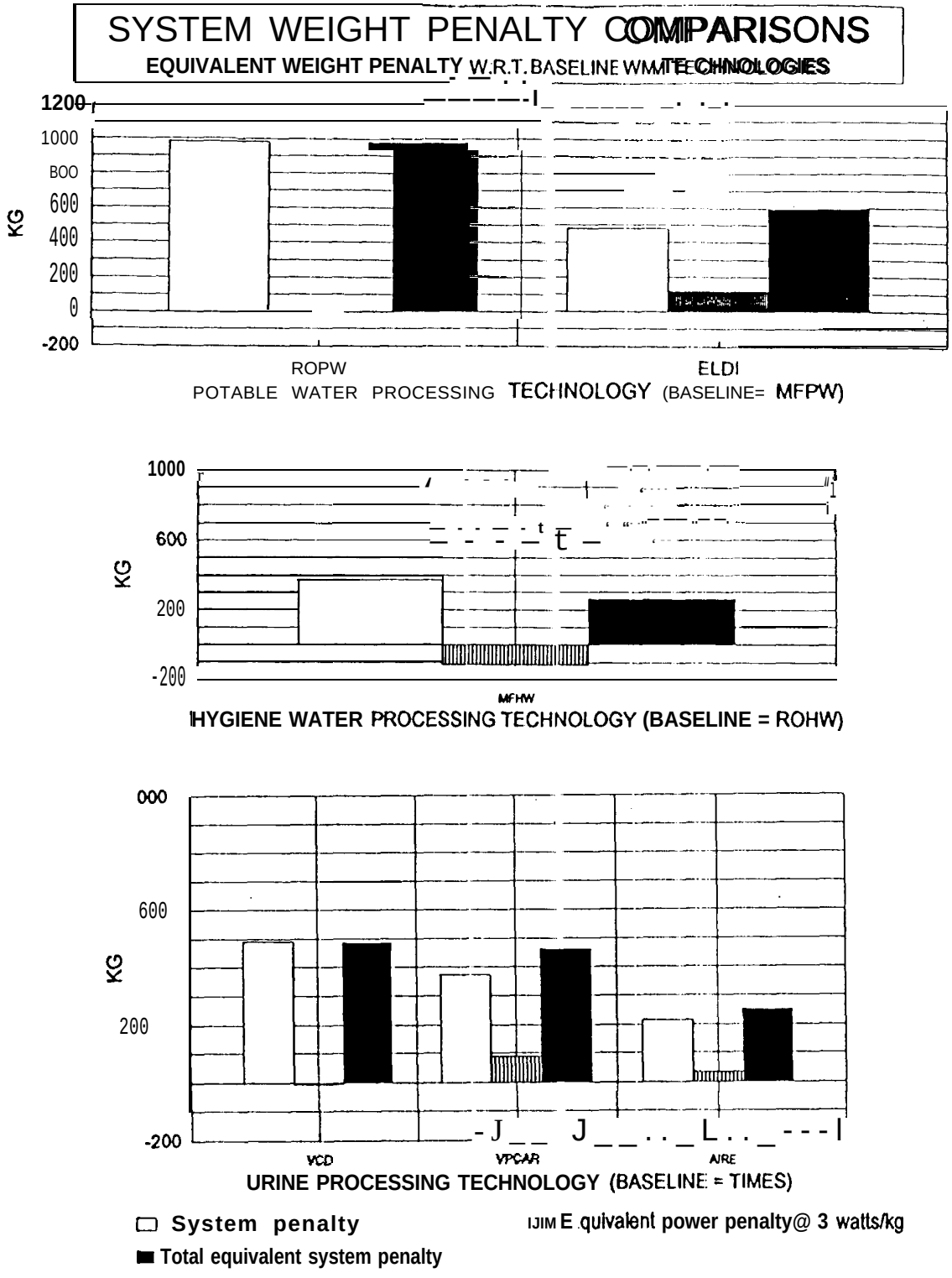


Figure V-22 . Equivalent System Weight Comparisons: Water Management Technologies

Solid Waste Treatment Technologies:

Figure V-23 shows the total equivalent system penalties for all of the SWT technologies. The drying processes (FD and HD) and the oxidation processes (COMB, WOX, SCWO) are compared to the baseline, which has no solid waste treatment.

SWT technologies show total equivalent penalties ranging from 350 kg to over 1050 kg. As discussed above, by introducing SWT processing, a surplus of clean water is produced; this surplus could be reduced by decreasing the amount of water in stored food. Power equivalent for the oxidation processes are similar (300 kg); however, due to the reported ability of SCWO to create near-potable quality water, system weight of the SCWO is lowest.

The weight advantage for SCWO is dependent upon the mission duration and the assumption that SCWO condensate does not require further treatment. At 90 days, the weight of the SCWO hardware dominates any weight advantage gained by producing clean condensate as shown in Figure V-24. At 90 days, the overall SCWO system weight penalty (excluding the equivalent power penalty) is 480 kg; when the mission length is increased to 600 days, the use of SCWO results in the penalty decreasing to 50 kg over the baseline as shown in Figure V-23. By increasing the mission duration to 700 days, the system weight penalty for SCWO goes to zero and becomes an advantage. However, the power penalty would still result in the SCWO having a total equivalent weight penalty of about 300 kg. In order for SCWO to have a weight advantage, mission lengths of about 1200 days for a crew size of four would be required. However, if it is assumed that the SCWO condensate requires additional cleanup before being accepted as either potable or hygiene water, then it is unlikely that any system weight advantages will be realized.

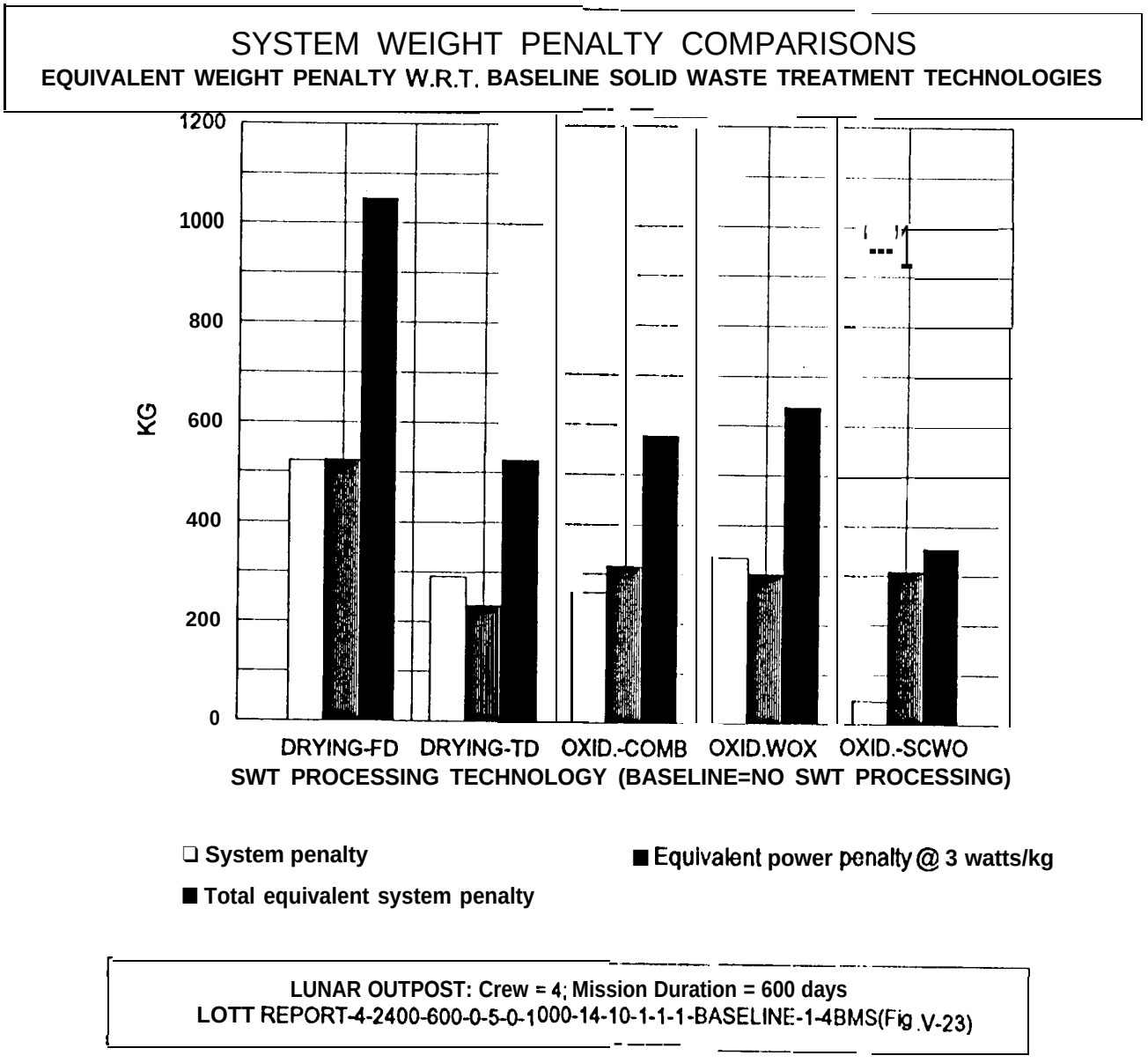


Figure V-23 . Equivalent System Weight Comparisons: Solid Waste Treatment Technologies

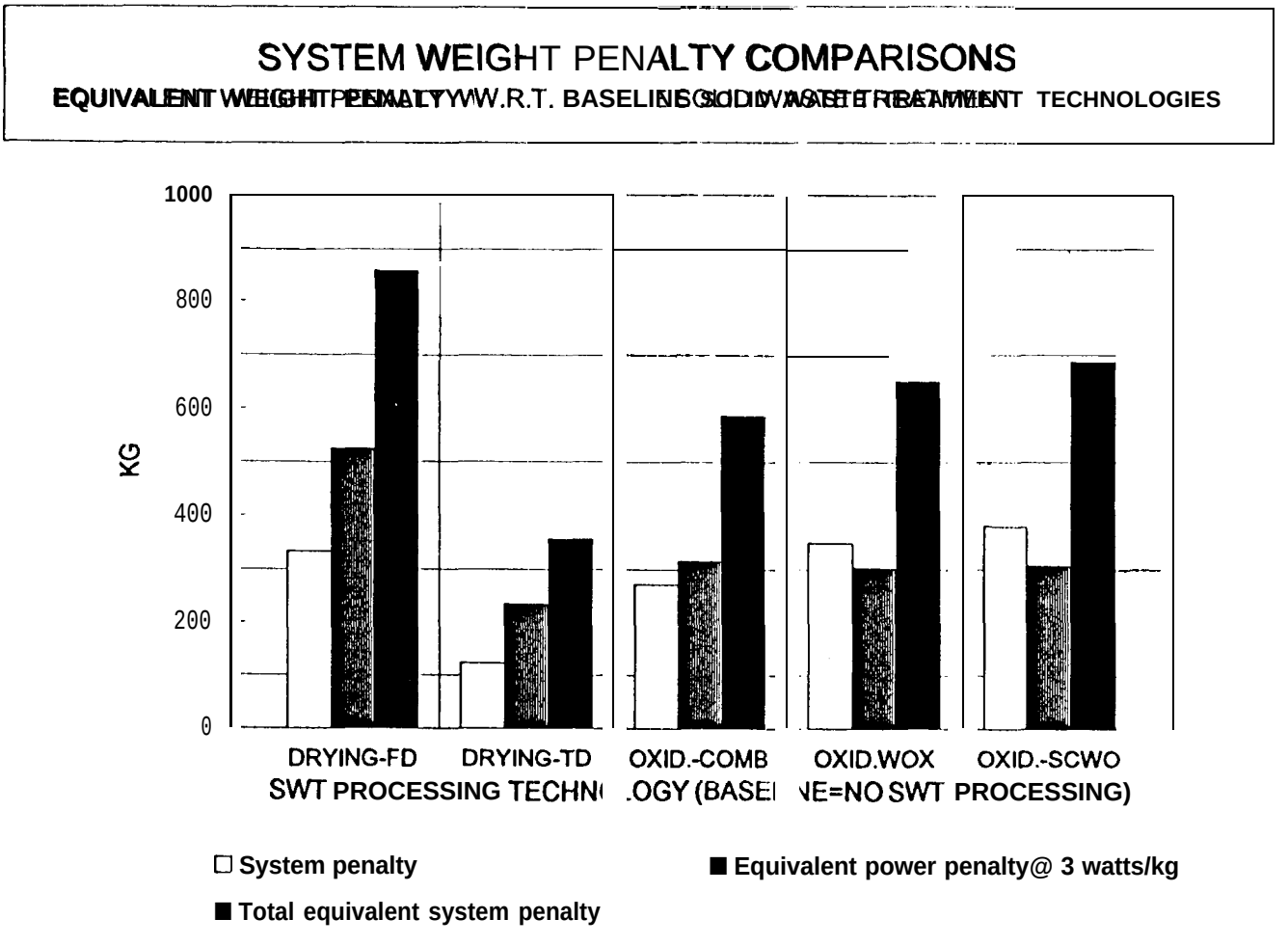


Figure V-24 . Equivalent System Weight Comparisons : Solid Waste Treatment Technologies (90 days)

VI. CONCLUSIONS AND RECOMMENDATIONS

1. Conclusions

As all of the regenerative technologies used in this study are still under development, conclusions concerning the best technologies must be revisited following significant progress in technology development. Hence, identifying a less-developed technology as having an advantage over a more-developed technology must be seen only as identifying a potential advantage that could be realized only by further investment in technology development. Some of the technologies are currently included in the design of the Space Station and therefore represent considerable technological maturity. Some technologies are conceptual in nature with varying degrees of uncertainty associated with the data collected on these technologies; the degree of data uncertainty is qualitatively represented by the validity level ranking.

A baseline set of technologies has been used against which comparisons have been made with a crew size of four. The baseline technologies are:

Air Revitalization Subsystem:	
CO_2 Removal:	Four Bed Molecular Sieve
CO_2 Reduction:	Bosch
O_2 Generation:	Static Feed Water Electrolysis
Water Management Subsystem:	
Potable Water Processing:	Multifiltration
Hygiene Water Processing:	Reverse Osmosis
Urine Processing:	Thermoelectric Integrated Membrane Evaporation System
Solid Waste Treatment Subsystem:	
Drying:	None
Oxidation:	None .

For the 21 regenerative cases run (Case 6, using lithium hydroxide is considered nonregenerative), overall regenerative system weights vary from a -5 to a +9 weight% relative to the baseline weight of 4060 kg for 90 days; for 600 days, the variation from the baseline weight of 13,920 kg varied from a -4 to a +10 weight%. For the nonregenerative case where lithium hydroxide was used for CO_2 removal, the system weight penalty was 7 weight% for 90 days and 32 weight% for 600 days.

Overall system power varied from a -8% to a +29% relative to the baseline power of 5470 watts (excluding the nonregenerative LiOH case) . When comparing only air revitalization and water management technologies, the variation narrows to -8 to +6%.

Regenerative technologies showing significant weight advantages include CO_2 electrolysis/Boudouard and water vapor electrolysis. Regenerative technologies showing significant power advantages include two bed molecular sieve, electrochemical depolarized concentrator, solid amine water resorption, co_2 electrolysis/ Boudouard, and multifiltration hygiene water.

When power is equated to equivalent weight (3 watts/kg for a regenerative fuel cell system) and added to the system weight, the two bed molecular sieve, co_2 electrolysis/Boudouard, and water vapor electrolysis have advantages over the baseline for long durations.

For mission durations below 700 days, there are no overall weight advantages realized by solid waste treatment processing. The decision to include solid waste treatment must therefore be based on considerations other than system weight reduction. For mission durations beyond 700 days, supercritical water oxidation technology is attractive relative to the baseline as it may produce a high quality water condensate. However, the high power and safety issues arising from the high pressure operation of the supercritical water oxidation must be balanced against its weight advantages. Total equivalent weight advantages of supercritical water oxidation relative to the baseline would require extremely long durations of over 1200 days.

Table VI-1 summarizes advantages, disadvantages, and validity levels of the technology choices for the CO_2 removal, CO_2 reduction, and O_2 generation functions.

Table VI-1. Comparisons of Air Revitalization Subsystem Technologies

SFE	TECHNOLOGY	ADVANTAGES	DISADVANTAGE	VALIDITY LEVEL
CO_2 Removal	FourBadMolecular Sieve	Maturity; Space Station development	SFE Power	3
" "	Two Bed Molecular Sieve	SFEweight SFE power	Maturity	4
" "	Electrochemical Depolarized Concentrator	SFE power	Effect on AR power Maturity H2 Required	4
" "	Air Polarized Concentrator		Maturity	7
" "	Solid Amine Water Resorption	SFE power	Effect on WM-consumables Maturity	7
" "	Lithium Hydroxide	SFE power; AR power; Maturity	Nonregenerative, consumables	3
CO_2 Reduction	Bosch	Carbon and oxygen recovery	Catalyst activity Consumable canister	3
" "	Sabatier	Maturity: Space Station development SFE simplicity	Effect on AR subsys High H_2 to CO_2 ratio	3
" "	Advanced Carbon Reactor System	Low consumables	Maturity 1 wo reactors, compl	4
" "	CO2 Electrolysis/ Boudouard	Produces oxygen; Low consumables due to WM subsystem effect; LowARPower	Maturity HighSFEpower	7
02 Generation	Static Feed Water Electrolysis	Maturiiy:Space Station development	High SFE power	3
"	Water Vapor Electrolysis	LowSFEand AR subsystem power; Low consumables due to WM subsystem effect	Maturity	7
" "	Solid Polymer Electrolyte	Stable long term cell activity; Maturity (submarines)	Slightly higher SFE	7

Note: SFE = Subsystem Functional Element
AR = Air Revitalization
WM = Water Management

Table VI-2 summarizes advantages, disadvantages and validity levels of the technology choices for the potable water processing, hygiene water processing, and urine processing functions.

Table VI-2. Comparisons of Water Management Subsystem Technologies

SFE	TECHNOLOGY	ADVANTAGES	DISADVANTAGES	VALIDITY LEVEL
Potable Water Processing	Multifiltration	H_2O recovery Maturity: Space Station development	Consumables	3
"	Reverse Osmosis	Low consumables; Maturity (H_2O desalinization)	H20 recovery	3
"	Electrochemical Deionization	H_2O recovery	Maturity; SFE Power	7
Hygiene Water Processing	Reverse Osmosis	Maturity: water desalinization	Power	3
"	Multifiltration	Maturity Space Station development; H_2O recovery	Consumables	3
Unne Processing	Thermoelectric Integrated Membrane Evaporation System	H_2O recovery	Maturity; Membrane fouling	3
"	Vapor Compression Distillation	Maturity: Space Station development	Complexity (mechanical)	3
"	Vapor Phase Catalytic Ammonia Removal	Volatiles treatment	H20 recovery; SFE power, Maturity	7
"	Air Evaporation	High H_2O recovery	Maturity; Consumables	7

Note: SFE = Subsystem Functional Element

Table VI-3 summarizes advantages, disadvantages and validity levels of the technology choices for-the drying and-oxidation functions within the solid waste treatment subsystem.

Table VI-3. Comparisons of Solid Waste Treatment Subsystem Technologies

SFE	TECHNOLOGY	ADVANTAGES	DISADVANTAGE S	VALIDITY LEVEL
Drying	Freeze Drying	Condensate quality; Maturity (other medical lab applications)	Maturity; SFE weight; SFE power; Unreacted solids disposal	7
"	Thermal Drying	Potential to use low grade heat	Condensate purity; Maturity; SFE weight; SFE power; Unreacted solids disposal	7
Oxidation	Combustion	Low pressure; Minimizes hazardous solids	Maturity; SFE weight; SFE power; High temperature; Condensate quality	7
"	Wet Oxidation	Maturity (other waste water applications); Minimizes hazardous solids	High pressure; Maturity; SFE weight; SFEpower	7
"	Super Critical Water Oxidation	Condensate quality; Minimizes WM consumables; Maturity (other waste water applications); Minimizes hazardous solids; Nearly complete organic destruction	High pressure; High temperature; Maturity; SFE weight; SFE power	7

Note: SFE = Subsystem Functional Element
AR = Air Revitalization
WM = Water Management

2. Recommendations

The following recommendations are based on the authors' observations not only during the performance of this study but also as the LiSSA tool was being developed:

1. *As technologies are funded for development, it is important to require contractors to generate and report data that can be utilized for quantitative technology comparisons.* Estimates of heat and material balances, equipment weights, power, volumes, and scaleup parameters should be a part of the technology development effort. It is suggested that NASA technical monitors add a "NASA Perspective" summary page to the final report such that any overly optimistic or conservative estimates or performances can be identified.

2. *In general, technology development directions should be aimed at reducing the weight of resupplies.* Nonregenerable supplies impose additional weight to be carried by a spacecraft plus additional manpower required for resupply operations.

3. *Technology development should be directed to outperform the current best technology or a carefully selected baseline technology.* Baseline technologies should be identified that have well documented weights, power usage, volume, feed and product characterizations, in addition to quantitative scaleup procedures.

4. *Basic research should be directed towards identification and use of lighter materials of construction, minimization or elimination of resupplies, and minimization of power demand.* Basic research is needed, for example, in the regeneration of sorption beds and membrane fouling for water purification, and Bosch carbon deposition kinetics and catalysts for air revitalization.

5. *The effects of process dynamics on technology trades should be examined.* Most of the processes Investigated do not operate in a continuous mode and must deal with fluctuating feed rates and compositions. However, processes that can be designed to be continuous tend to be lighter and energy efficient. If the dynamics of the process and the fluctuating feed rates and compositions can be modeled so that effective control strategies are possible, the advantages of a continuous process design can be realized.

6. *Systems analysis is an iterative and continuing process throughout the technology development eye-l e from concept evaluation to mission readiness.* By stepping back again and again to obtain a system view following technology selections for further development. or mission system design, systems analysis enables significant cost reductions in developing, designing and commissioning any complex system. LiSSA is such an analysis tool for physical-chemical life support systems.

7. *Life support systems analysis should be extended to include biological systems and in situ resource utilization systems so that technologies pertaining to these systems can be traded for assessment of system impacts.* The modular and architectural construction of LiSSA lends itself to performing these trades [Reference ES-l]. In addition, future trades should include power and propulsion systems to complete the picture for mission and project planners.

This page left intentionally blank

VII. REFERENCES

ES-1 . Ganapathi, G. B., Seshan, P. K., Rohatgi, N. K. ,and Ferrall, J. F., "Human Life Support During Interplanetary Travel and Domicile - Part VI: Hybrid P/C - Generic Modular Flow Schematic for Biological Life Support Systems, " SAE paper 921120, presented at the 22nd ICES Conference, July 1992.

1-1. Ferrall, J.F., Rohatgi, N.K., and Seshan, P.K., Project Pathfinder - Physical Chemical Closed-loop Life Support - Systems Analysis and Assessment - FY'89 Annual Report, JPL D-12401 (internal report), Jet Propulsion Laboratory, Pasadena, CA, January 1990.

I-2. Seshan, P.K., Ferrall, J., and Rohatgi, N., "Human Life Support During Interplanetary Travel and Domicile - Part I: System Approach, " SAE paper 891431, presented at the 19th ICES Conference, July 1989.

I-3. Ferrall, J., Seshan, P.K., and Rohatgi, N., "Human Life Support During Interplanetary Travel and Domicile - Part II: Generic Modular Flow Schematic Modeling," SAE paper 911322, presented at the 21st ICES Conference, July 1991.

I-4. Seshan, P.K., Ferrall, J., and Rohatgi, N., "Human Life Support During Interplanetary Travel and Domicile - Part IV: Mars Expedition System Trade Study," SAE paper 911323, presented at the 21st ICES Conference, July 1991.

I-5. Rohatgi, N. K. , Ferrall, J., and Seshan, P.K., "Human Life Support During Interplanetary Travel and Domicile - Part IV: Mars Expedition Technology Trade Study," SAE paper 911324, presented at the 21st ICES Conference, July 1991..

I-6. Ferrall, J., Rohatgi, N. K. ,and Seshan, P.K., "Human Life Support During Interplanetary Travel and Domicile -- Part V: Mars Expedition Technology Trade Study for Solid Waste Management, " SAE paper 921119, presented at the 22nd ICES Conference, July 1992.

I-7. Ferrall, J. Seshan, P.K., Rohatgi, N.K., and Ganapathi, G. B., "Generic Modeling of a Life Support System for Process Technology Comparisons, " Session/paper number 101b, presented at the March 1993 American Institute of Chemical Engineers Spring National Meeting, Houston, TX, March 1993.

I-8. Ferrall, J., Rohatgi,N.K., Ganapathi, G.B., and Seshan, P.K., LiSSA-ST User Manual, JPL D-11163 (internal report), Jet Propulsion Laboratory, Pasadena, CA, September 1993.

I-9. Ferrall, J., Rohatgi, N.K., Ganapathi, G.B., and Seshan, P. K., LiSSA-TT User Manual, JPL D-11164 (internal report.), Jet Propulsion Laboratory, Pasadena, CA, September 1993.

1-10. Ferrall, J., Rohatgi, N.K., Ganapathi, G.B., and Seshan, P.K., LiSSA-ST and LiSSA-TT User Manuals, JPL D-11165' (internal report), Jet Propulsion Laboratory, Pasadena, CA, September 1993.

II-1. Environmental Control and Life Support System Architectural Control Document, SSP 30262, Revision D, NASA Marshall Space Flight Center, Huntsville, AL, July 1991.

II-2. Wydeven, T. and Golub, M., Generations Rates and Chemical Compositions of Waste Streams in a Typical Crewed Space Habitat, NASA TM-102799, NASA Ames Research Center, Moffett Field, CA, 1990.

II-3. Golub, M., and Wydeven, T., Waste Streams in a Typical Crewed Space Habitat: An Update, NASA 'J'M-103888, NASA Ames Research Center, Moffett Field, CA, February 1992.

II-4. Volk, T., and Rummel, J.D., "Mass Balances for a Biological Life Support System Simulation Model," COSPAR Proceedings, 1986.

IV-1. Rohatgi, N., Ballin, M.G., Seshan, P.K., Bilardo, V.J., and Ferrall, J., "Hardware Scaleup Procedures for P/C Life Support Systems," SAE paper 911395, presented at the 21st ICES Conference, July 1991.

IV-2. Space Station ECLSS Evolution Study Report to MSFC, McDonnell Douglas, Huntsville, Alabama, NAS8-36407, June 30, 1989.

IV-3. Rohatgi, N.R., Ferrall, J.F., and Seshan, P.K., "Extracted Data Taken by JPL From the Critical Design Review And Preliminary Design Review Data Packages for the Technology Demonstration Program Of Space Station Freedom - Data Presented by Subcontractors to Boeing Aerospace - 1987," JPL D-12402 (internal report), Jet Propulsion Laboratory, Pasadena, CA, January 1991.

IV-4. Rohatgi, N.R., Ferrall, J.F., and Seshan, P.k.,"Notes Taken by JPL from Presentation Made by Dr. C. H. Lin (NASA Johnson Space Center, Houston, TX) - April 1992," JPL D-12400 (internal report), Jet Propulsion Laboratory, Pasadena, CA, January 1991.

IV-5. Component Data Handbook - Environmental Control and Life Support Systems, Hamilton Standard Division of United Technologies Corp., 1980.

IV-6. A Guide t-o Freeze Drying for the Laboratory, Labconco Corporation, An Industry Service Publication, Kansas City, Missouri, 1987.

IV-7 . Fisher Scientific Catalog, 1988.

IV-8. Labak, L.J., Remus, G.A., and Shapira,J.,"Dry Incineration of Wastes for Aerospace Waste Management Systems, " ASME 72-ENAV-2, 1972.

IV-9. Slavin, T., Liening, F., Oleson, M., and Olson, R.L., Controlled Ecological Life Support Systems) Physiochemical Waste Management Systems Evaluation, NASA Contractor Report 177422, prepared by Boeing Aerospace Company, June 1986.

IV-10. Jagow, R., Jaffe, R., and Saunders, C., "The Processing of Human Wastes by Wet Oxidation for Manned Spacecraft", ASME 70-AV/Sp T-1, 1970.

IV-11. Hong, G., and Fowler, P., Supercritical Water Oxidation of Urine and Feces, final report prepared by MODAR, Inc. for NASA-JSC under Contract NAS2-12176, 1987.

IV-12. Standard Handbook of Hazardous Waste Treatment and Disposal, Freeman, H.M., editor, McGraw-Hill. Book Company, 1989.

v-l. Stafford,T., America at the Threshold-Report of the Synthesis Group on America's Space Exploration Initiative, prepared by the Synthesis Group for Vice-President J. D. Quayle, Chairman of the National Space Council, May 1991.

This page left intentionally blank

APPENDIX A

DESCRIPTION OF LiSSA TOOL

The potential complexity of future life support systems for manned missions necessitates the development of the appropriate systems analysis capability within NASA as a guide to technology and systems development (Evanich et al., 1991) . The life support system (LSS) most appropriate for a given human exploration of outer space must be chosen from candidates ranging from a very simple, nonregenerative LSS to a very complex, integrated physical-chemical, and possibly biological, closed-loop LSS. There are many regenerative processes that are potential candidates to provide a particular function as part of the overall LSS. To synthesize an LSS, all of the processes must be integrated to perform certain generic life support functions such as air revitalization and water recovery.

A GMFS architecture has been developed to enable synthesis, analysis, and eventual selection of system and technology options for defined missions. The architecture consists of a modular, top-down hierarchical break-down of the physical-chemical closed loop life support (P/C CLLS) system into subsystems, and a further break-down of subsystems into subsystem functional elements (SFEs) representing individual processing technologies. This approach allows for modular substitution of technologies and subsystems and for the traceability of parameters through all the hierarchical levels, which is useful in comparing systems or technologies rapidly and accurately. The GMFS is the central feature utilized by the Life Support Systems Analysis (LiSSA) tool created by JPL as illustrated in Figure A-1.

A series of papers, describing the technique and results, titled "Human Life Support During Interplanetary Travel and Domicile" (Parts I,II,III, IV, and V), have been presented at recent International Conference on Environmental Systems (ICES) meetings. (It should be noted that the acronym LiSSA was adopted in early 1992 and therefore will not be found in earlier papers.) Another paper presented at the 21st ICES conference described hardware scaleup procedures used in the LiSSA trade tool (Rohatgi et al., 1991a). A paper was presented at the March 1993 American Institute of Chemical Engineers meeting that illustrated how the tool can be utilized to do technology trades and system optimization investigations.

LiSSA APPROACH AND CALCULATION SCHEME

A schematic of the LiSSA methodology is given in Figure A-1. To initiate the analysis, the system matrix, technology matrix, system specifications, and mission specifications are first chosen.

The system matrix includes the types of life-support systems that are of interest. It could include non-waste-processing, open-loop systems, systems that process cabin air for carbon dioxide removal only, and closed-loop systems with varying degrees of closure of the oxygen and water loops. "Closing the loops" for oxygen and water is accomplished by processes that regenerate pure oxygen and clean water from waste streams generated by the crew. The amounts of oxygen and water regenerated depend on the efficiency of the regeneration processes selected for the system.

The technology matrix includes the processing technologies that would be utilized to regenerate oxygen and water. From this matrix, a baseline set of technologies can be chosen for configuring the various systems in the system matrix. Currently, this includes technologies under consideration for Space Station Freedom (SSF) and some additional advanced technologies.

System specifications include metabolic and hygiene inputs and outputs pertaining to the crew. These specifications are required as input parameters to the GMFS module integration and computer simulation. Mission specifications are required as parametric inputs to the LiSSA Trade Tool.

For all the technology candidates considered, performance data must be acquired and utilized to model technologies as modules using the ASPEN PLUS chemical process simulation package. Once all the ASPEN PLUS modules are written, they are stored in an insert library. The modules are integrated into the GMFS architecture by calling them from the library using insert statements in the ASPEN input file. The complete input-code package represents the LiSSA Simulation Tool to produce output as an American Standard Code for Information Interchange (ASCII) file (with the *.PRN extension) that is used as input to the LiSSA Trade Tool.

The link between the LiSSA Simulation Tool and the LiSSA Trade Tool is accomplished by a spreadsheet macro which processes and loads the ASCII file from the simulation output into the Trade Tool. The Trade Tool uses simulation output, mission specifications, and JPL-developed scaleup formulas for weight, power, and volume. The entire spreadsheet represents the systems analysis output with a variety of tables and graphs.

LiSSA CALCULATION SCHEME

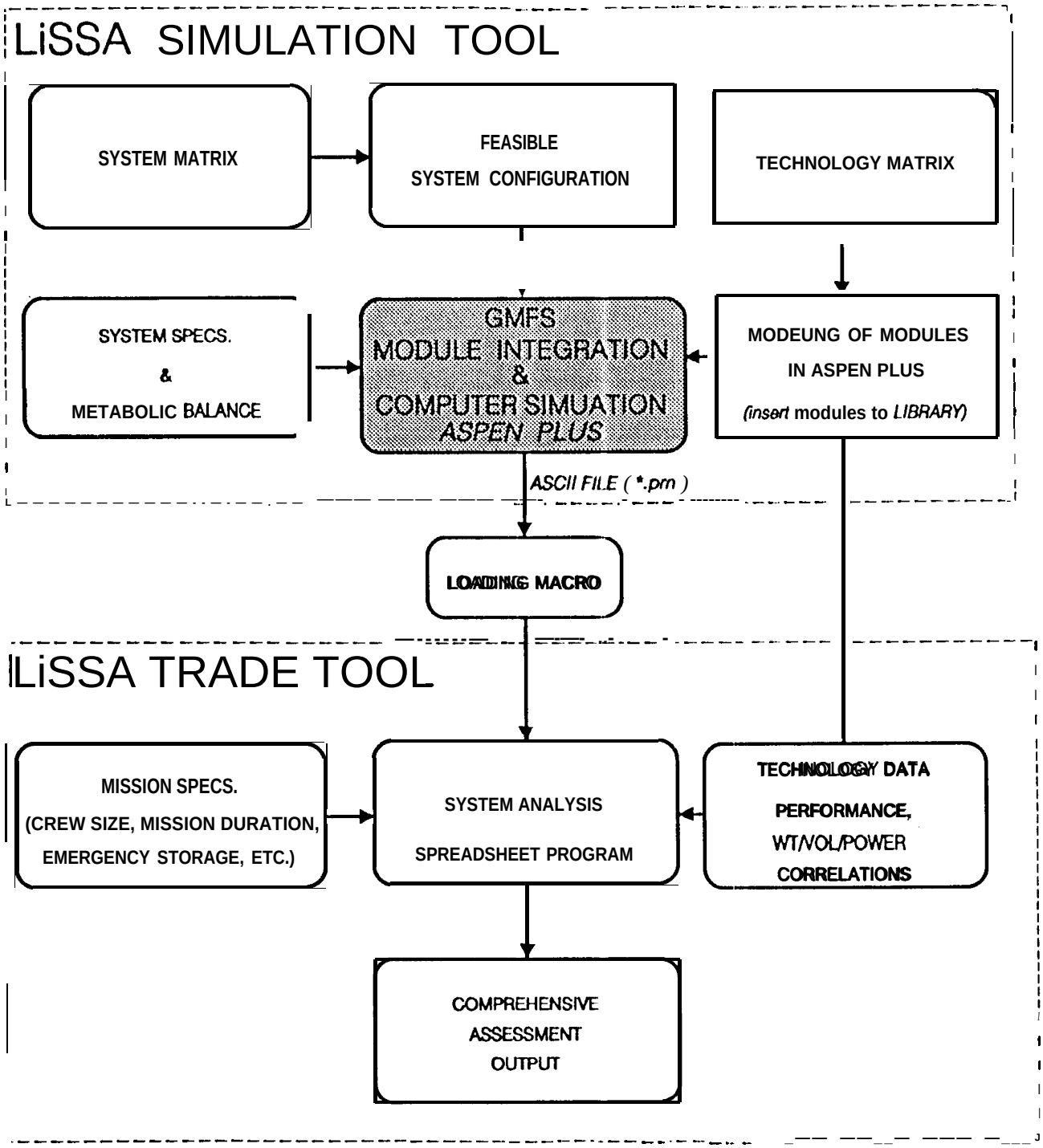


Figure A-1 . LiSSA Methodology

REFERENCES FOR APPENDIX A

Evanich, P. L., Voecks, G. V., and Seshan, P. K., "Advanced Life Support Technology Development for the Space Exploration Initiative, " AIAA paper AIAA-90-3726, presented at the AIAA Space Programs and Technologies Conference, Huntsville, AL, 'September 25-28, 1990.

Ferrall, J.F., Seshan, P.K., and Rohatgi, N.K. and Ganapathi, G.B., "Generic Modeling of a Life Support System for Process Technology Comparisons, " Session/paper number 101b, presented at the March 1993 AIChE Spring National Meeting, Houston, TX, March 1993.

Ferrall, J.F., Seshan, P.K., and Rohatgi, N.K., "Human Life Support During Interplanetary Travel and Domicile - Part II: Generic Modular Flow Schematic Modeling, " SAE paper 911322, presented at the 21st ICES Conference, July 1991 .

Rohatgi, N., Ballin, M.G., Seshan, P.K., Bilardo, V.J., and Ferrall, J., "Hardware Scaleup Procedures for P/C Life Support Systems, " SAE paper 911395, presented at the 21st ICES Conference, July 1991a.

Rohatgi, N.K., Seshan, P.K., and Ferrall, J.F., "Human Life Support During Interplanetary Travel and Domicile - Part IV: Mars Expedition Technology Trade Study," SAE paper 911324, presented at the 21st ICES Conference, July 1991b.

Seshan, P.K., Ferrall, J.F., and Rohatgi, N.K., "Human Life Support During Interplanetary 'I'ravel and Domicile - Part I: System Approach, " SAE paper 891431, presented at. the 19th ICES Conference, July 1989.

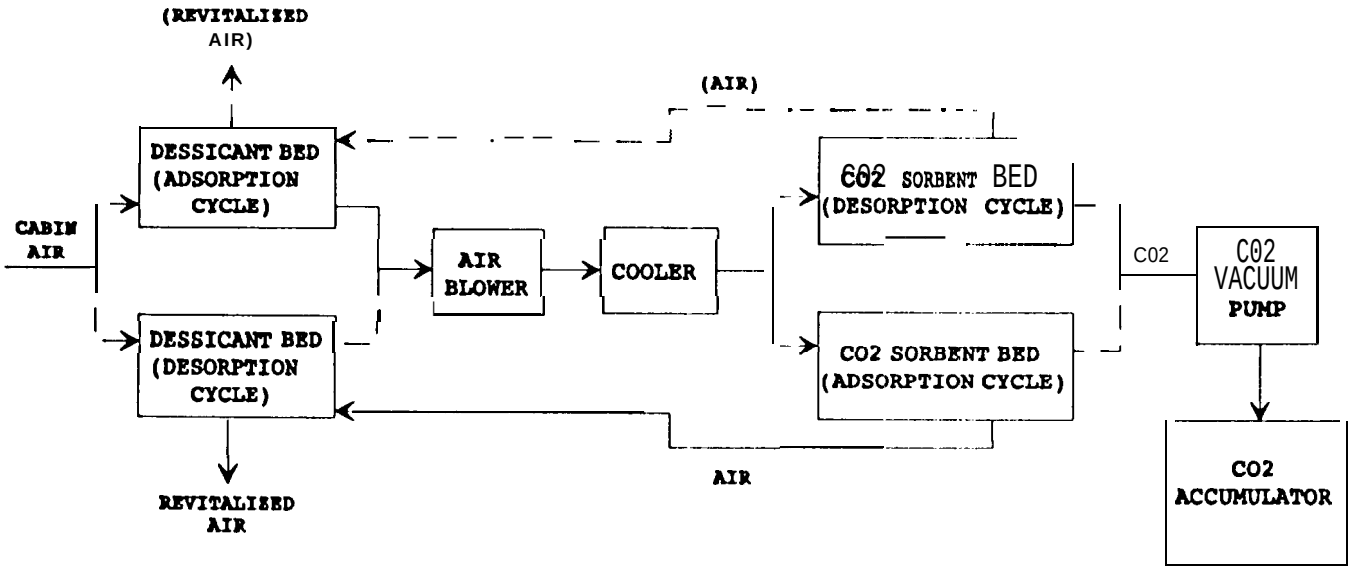
Seshan, P.K., Ferrall, J.F., and Rohatgi, N.K., "Human Life Support During Interplanetary Travel and Domicile - Part IV: Mars Expedition System Trade Study," SAE paper 911323, presented at the 21st ICES Conference, July 1991.

APPENDIX B

DESCRIPTIONS AND PROCESS FLOW SCHEMATICS

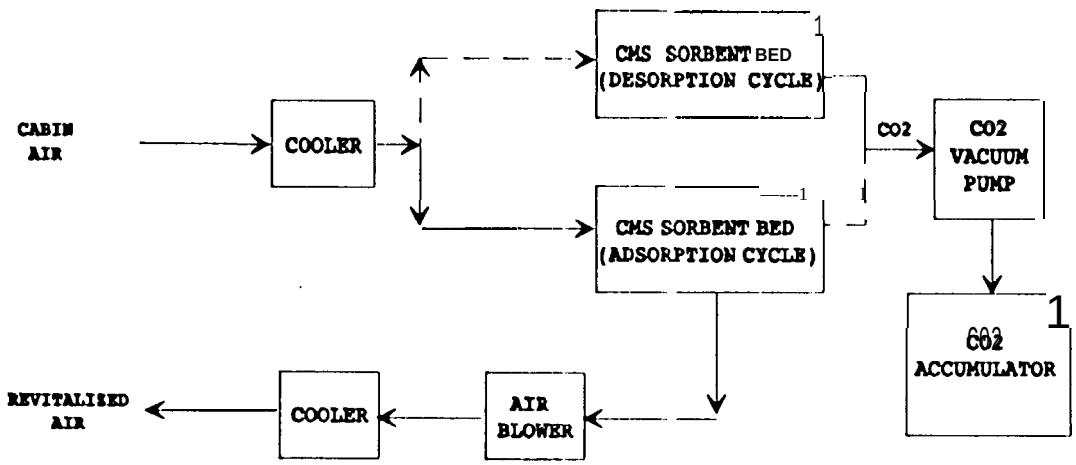
OF PHYSICAL/CHEMICAL LIFE SUPPORT TECHNOLOGIES

The Subsystem Functional Element (SFE) functions included in this Appendix are CO_2 removal, CO_2 reduction, and O_2 generation for the air revitalization (AR) subsystem; potable water (PW) processing, hygiene water (HW) processing, and urine processing for the water management (WM) subsystem; and drying and oxidation for the solid waste treatment (SWT) subsystem. Data sources for technologies are given in Tables IV-2, IV-3, and IV-4. Functional schematics and brief descriptions of the technologies used for the trades presented in the report are included.



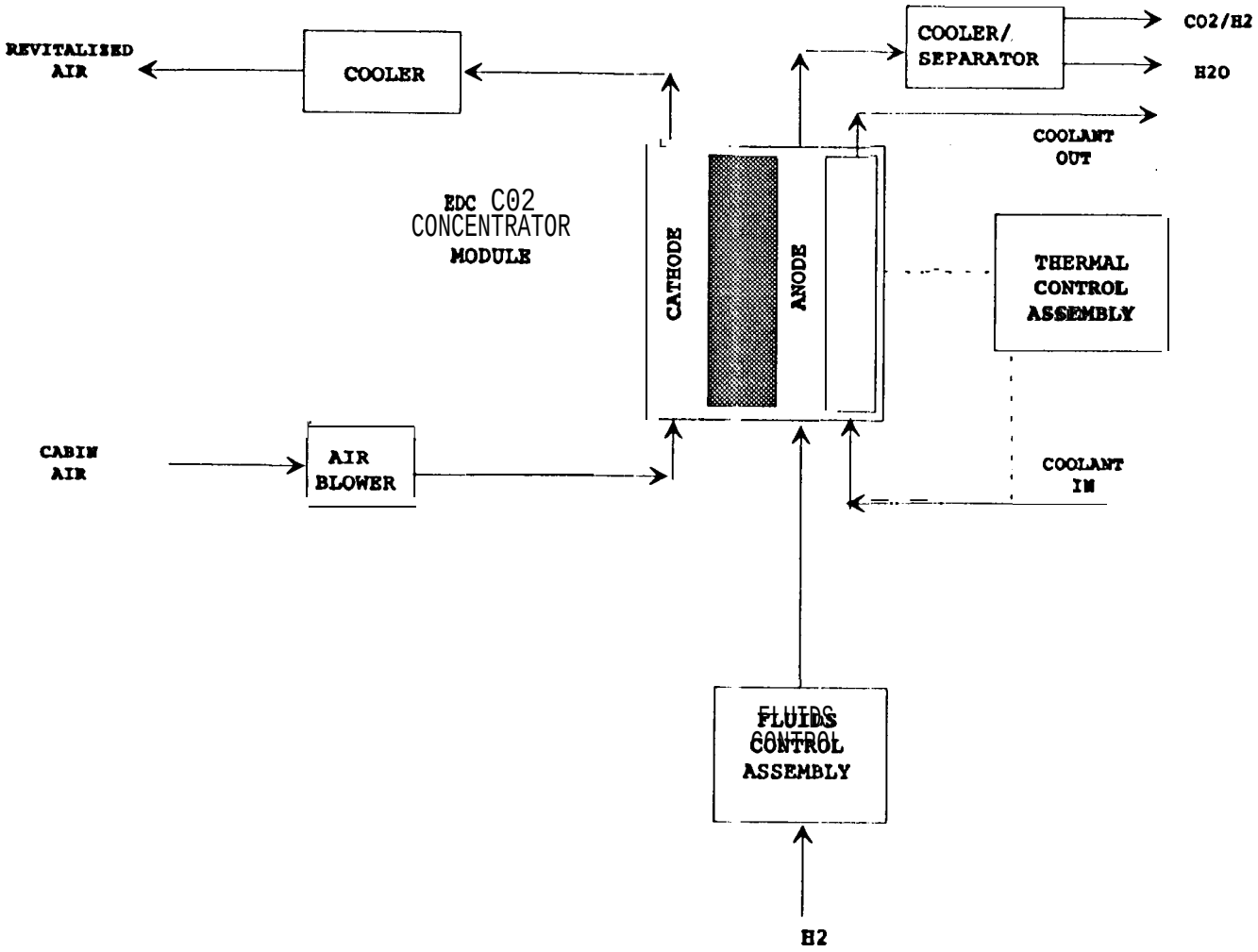
Process Flow Schematic for Four-Bed Molecular Sieve

The Four-Bed Molecular Sieve(4BMS) removes CO_2 from the inlet air stream via an adsorption process. Water is removed from the inlet air stream in an adsorbent bed packed with a mixture of silica gel and zeolite 13x. The dry air stream is then cooled and fed to a CO_2 adsorbent bed packed with zeolite 5A. Additionally, previously adsorbed water and $C0_2$ sorbent beds are in a resorption cycle. Desorbed water is used to rehydrate processed air, and desorbed CO_2 is pumped to an accumulation tank. Dotted lines demonstrate flow for adsorption/desorption cycling initiated when the adsorption capacity of a bed has been reached.



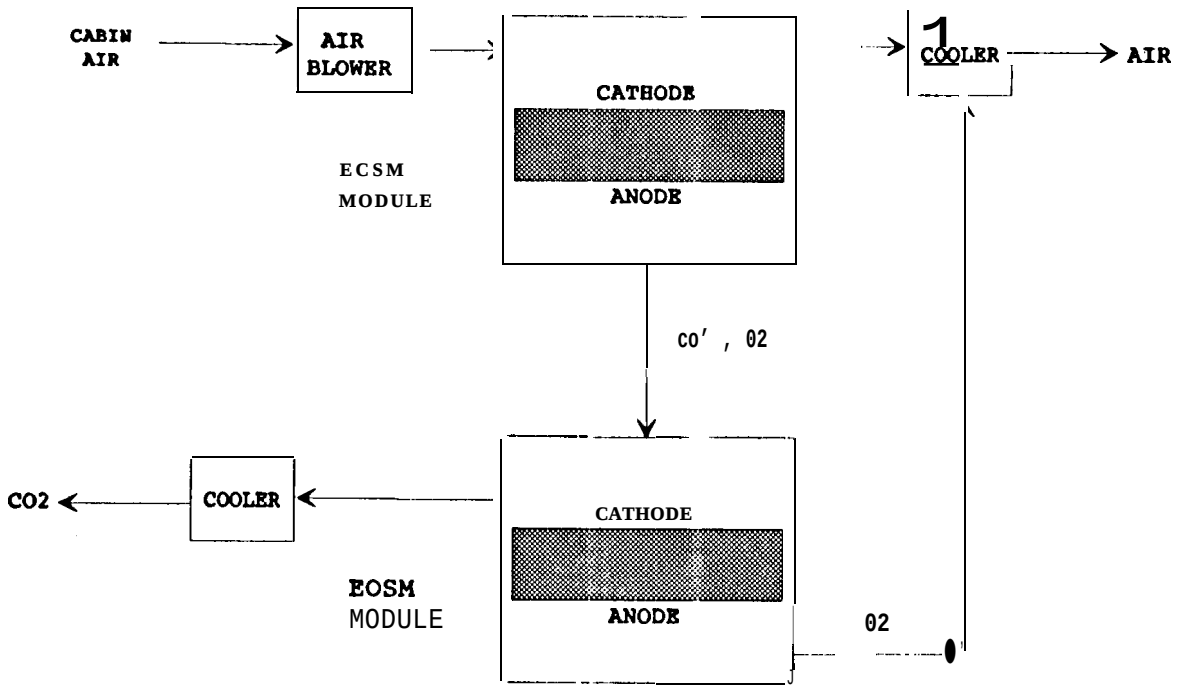
Process Flow Schematic for Two-Bed Molecular Sieve

The Two-Bed Molecular Sieve (2BMS) removes CO_2 from the inlet air stream via an adsorption process using a carbon molecular sieve (CMS). Unlike the zeolites of the 4BMS, the CMS is not affected by the moisture in the process stream. The 2BMS eliminates the requirement of desiccant beds; in addition, it also desorbs at a lower temperature than zeolites, thereby reducing regenerating power requirements. Dotted lines demonstrate flow for adsorption/desorption cycling initiated when the adsorption capacity of a bed has been reached.



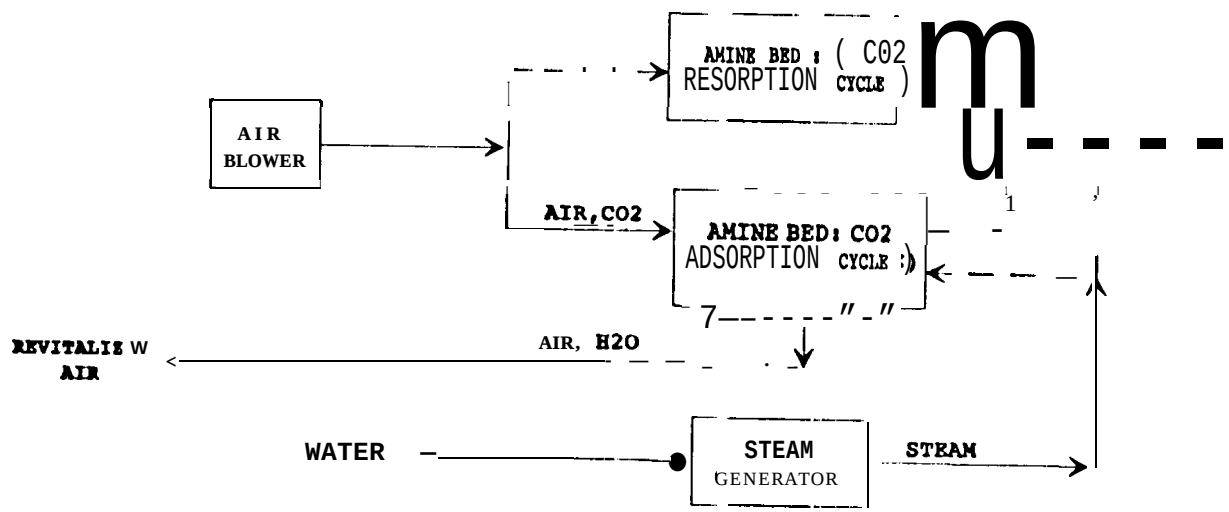
Process Flow Schematic for Electrochemical Depolarized CO_2 Concentrator

The Electrochemical Depolarized CO_2 Concentrator (ED) treats cabin air in an electrochemical cell. Air containing CO_2 passes through the cathode of an electrochemical cell utilizing an aqueous electrolyte. The CO_2 diffuses to the electrolyte-air interface where it is absorbed and reacted with hydroxyl (OH) ions to form carbonate (CO_3) and bicarbonate (HCO_3) ions. The carbonate and bicarbonate ions migrate to the cathode where $C0_2$ is released. When H_2 is supplied to the anode side, H_2O is also released; heat and electrical power are generated by the cell. The process requires a blower, fluids control assembly, and a thermal control assembly to remove heat from the cell.



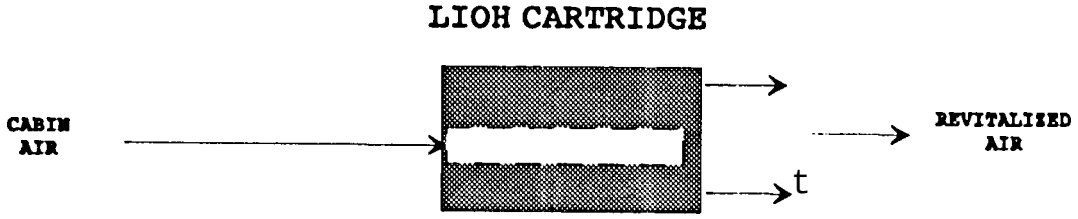
Process Flow Schematic for Air Polarized CO_2 Concentrator

The Air Polarized CO_2 Concentrator (APC) combines an electrochemical CO_2 separation module (ECSM) and an electrochemical O_2 separation module (EOSM) to remove CO_2 from cabin air. The ECSM is similar to the electrochemical cell used in the ED process; CO_2 diffuses to the electrolyte-air interface where it is absorbed and reacted with hydroxyl (OH) ions to form carbonate (CO3) and bicarbonate(HCO3) ions. The carbonate and bicarbonate ions migrate to the cathode where CO_2 is released. However, H_2 is not supplied to the ABC process; some of the O_2 in the air migrates via the electrochemical process to the anode where it is evolved with the $C0_2$. The O_2 and CO_2 are fed to the EOSM to remove most of the O_2 from the CO_2 stream using an acid-electrolyte cell. The process requires a blower, fluids control assembly, and a thermal control assembly to remove heat similar to the ED process.



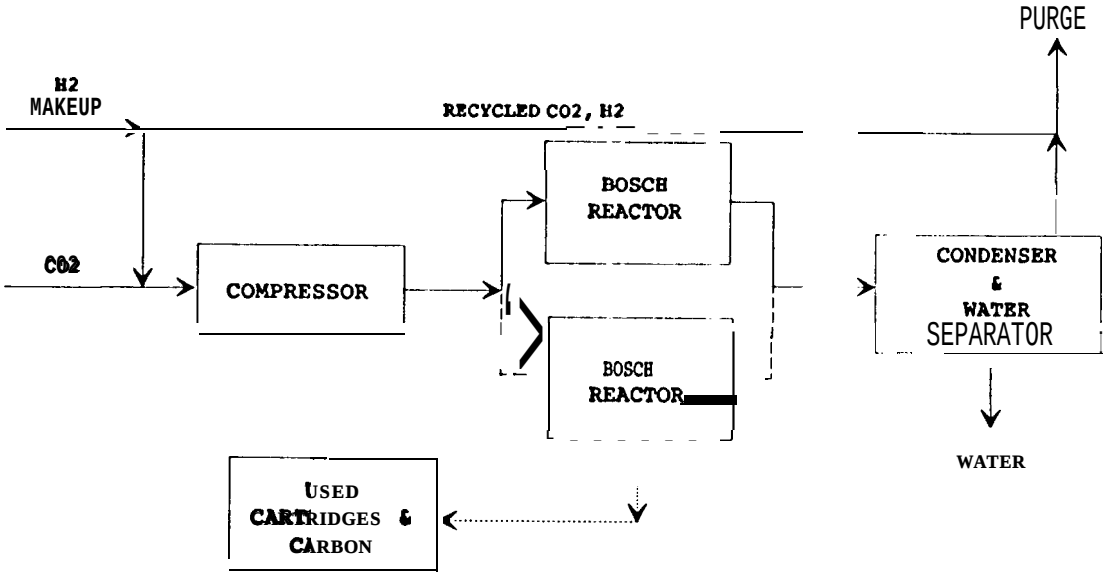
Process Flow Schematic for Solid Amine Water Resorption

The Solid Amine Water Resorption process (SAWD) removes CO_2 from the inlet air stream via an adsorption process. Dotted lines demonstrate flow for adsorption/resorption cycling initiated when the absorption capacity of a bed has been reached. Steam is used to desorb the CO_2 from the amine bed. During CO_2 absorption, the $C0_2$ replaces the adsorbed H_20 from the previous resorption cycle; the water removed from the bed places an additional load on the temperature and humidity control subsystem as it must condense the water vapor. Regeneration can take place at cabin pressure; i.e., vacuum conditions are not required.



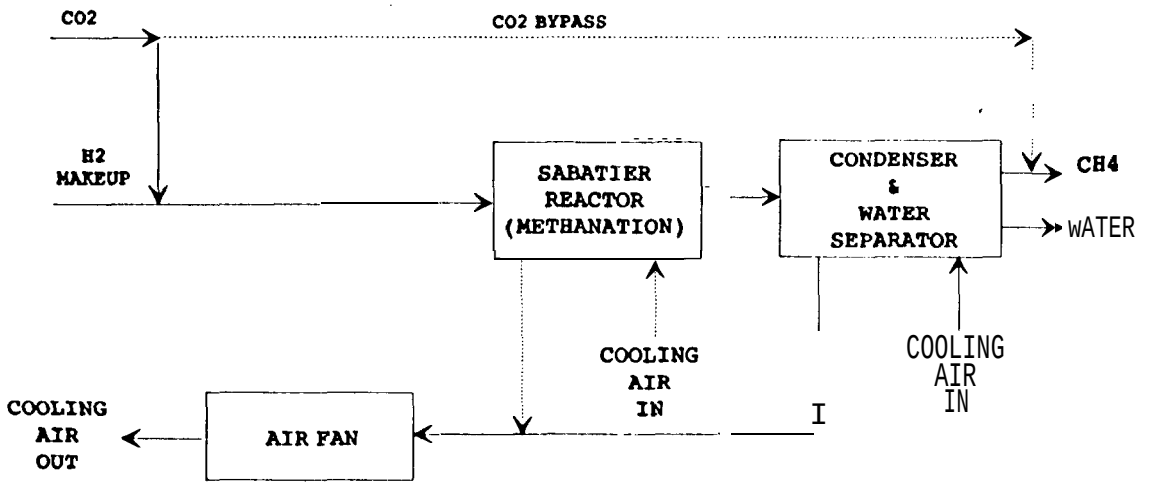
Process Flow Schematic for Lithium Hydroxide CO_2 Removal

This process uses a nonregenerable LiOH cartridge to remove CO_2. The cartridge consists of a radial flow cylindrical cartridge containing LiOH which is designed for ease of replacement after the absorber capacity has been reached. The cartridge also contains activated charcoal to control trace contaminant constituents in the cabin atmosphere. Cabin air enters the canister through a center tube and flows radially from the center through the charcoal bed where odor is removed, then through the LiOH, and finally through a particulate filter for dust removal before exiting the canister. Efficient absorption of CO_2 involves an initial H_2O absorption to form lithium hydroxide monohydrate ($LiOH-H_2O$); this is followed by absorption of CO_2 by the monohydrate forming the final carbonate (Li_2CO_3) and releasing H20. The overall process actually is a net producer of H_20 and heat.



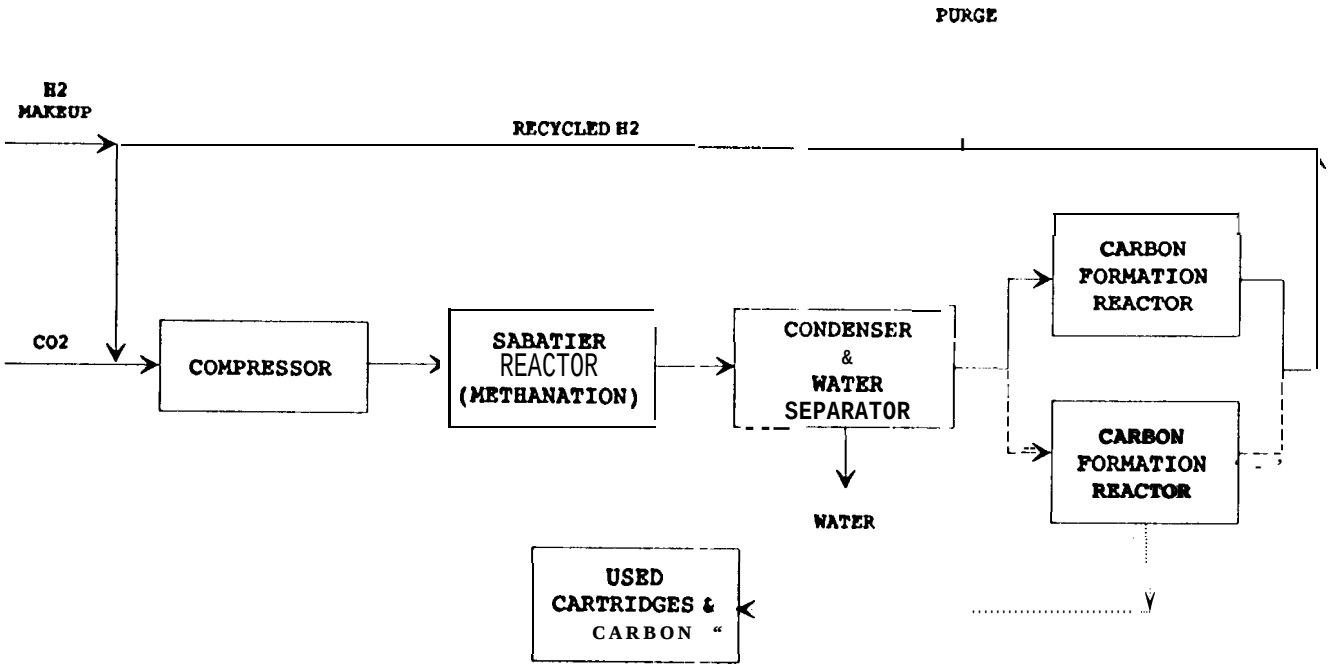
Process Flow Schematic for Bosch Reactor

The Bosch process reacts CO_2 with hydrogen in the presence of a steel wool catalyst to produce solid carbon and potable water. Less than 10% of the input CO_2 is reduced with a single pass through the Bosch reactor, but 100% conversion can be obtained by recirculating the process gases with continuous deposition of carbon and removal of water. CO_2 is directly reduced to carbon and water at 650°C in an expendable cartridge with iron catalyst. Two such reactors are required to maintain continuous operation and allow for cartridge replacement.



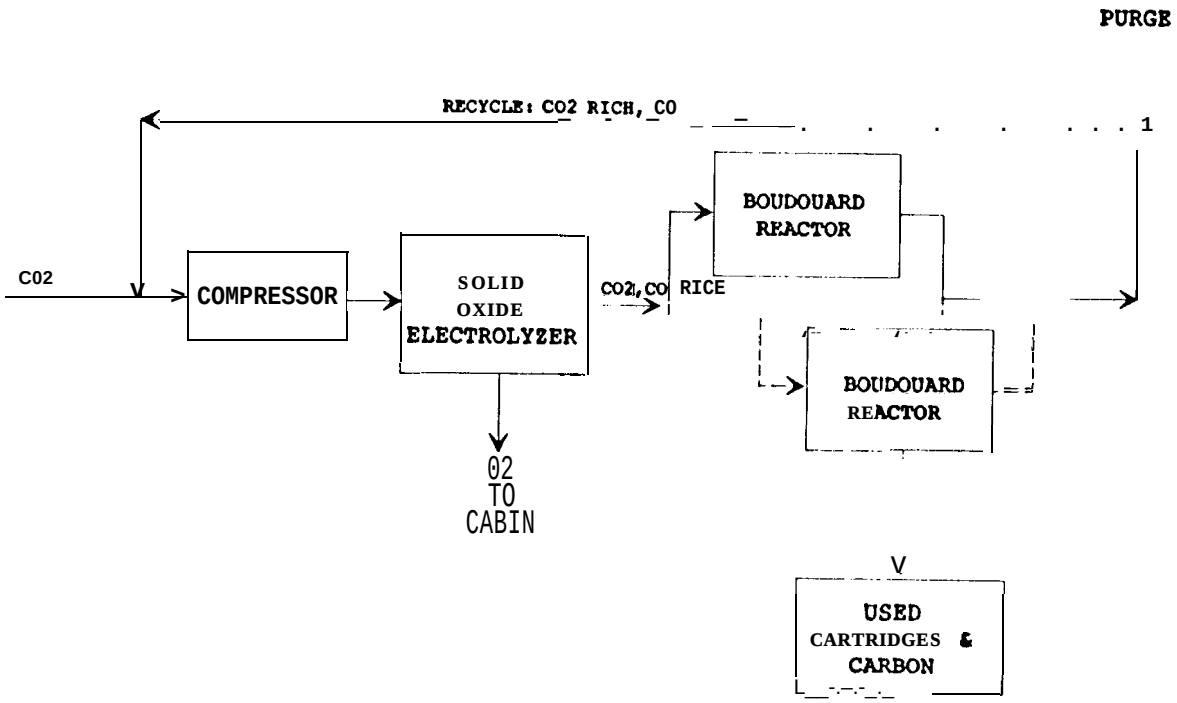
Process Flow Schematic: for Sabatier Reactor

CO_2 is methanated with H_2O at temperatures from 180°C to 530°C in the presence of a ruthenium catalyst on a granular substrate. The reactor produces CH4 and H_2O with a stoichiometric reactor feed ratio of 4 moles H_2 to 1 mole of CO_2. The reactor itself is equipped with electric start up heaters. 'l'he methanation reaction is exothermic; reactor feed gas enters one end of the reactor and flows down a central tube where it is regeneratively heated by the reactor product gases. The reactor is designed so that the feed gases flow back down the catalyst bed which is located in the annulus between the center tube and reactor housing. The reactor is designed to create a favorable temperature profile with high temperatures in the catalyst bed inlet (260° to 430°C) and lower temperatures in the outlet (90° to 260"C) . The gases leave the reactor between 90° and 150°C and are cooled to condense and separate the H_2O vapor product. The reactor includes air cooling to prevent overheating at elevated CO_2 reduction rates.



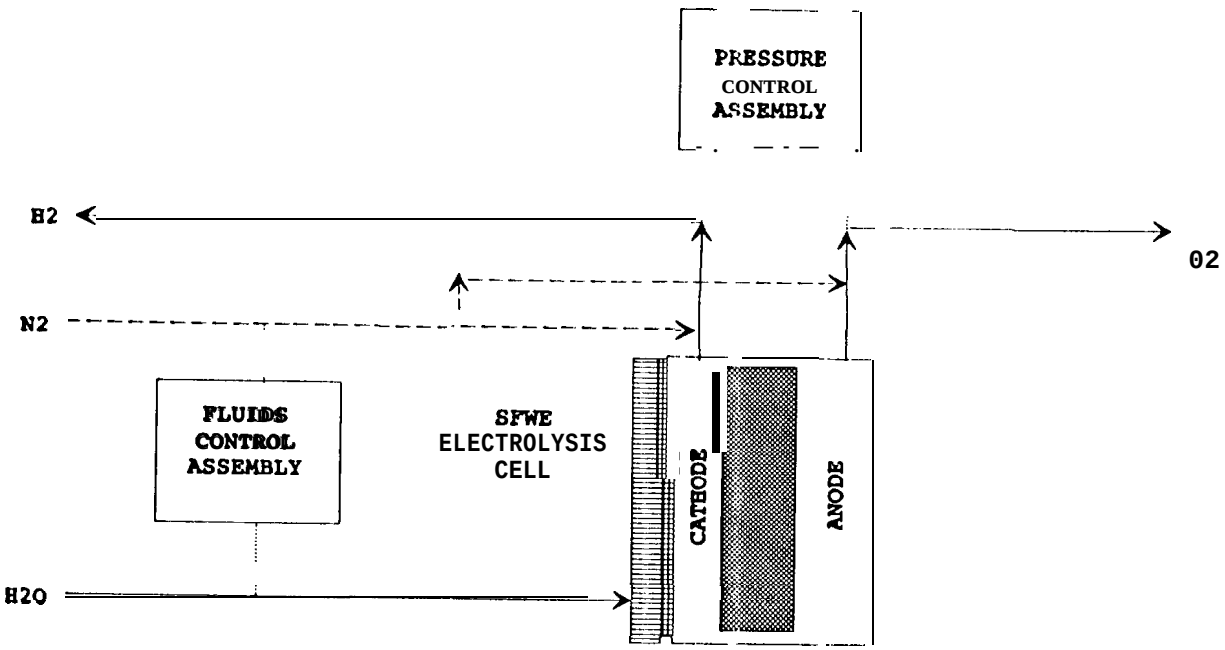
Process Flow Schematic for Advanced Carbon Reactor System

The Advanced Carbon Reactor (ACR) system consists of a Sabatier reactor, a gas/liquid separator to remove product water from methane, and a carbon formation reactor (CFR) to reduce methane to carbon and hydrogen. In the Sabatier reactor CO_2 is reacted with hydrogen in the presence of a ruthenium catalyst on a granular substrate. Operating temperatures range from 100° to 600°C, and reactor efficiency is greater than 98%. Water from the produce stream is then removed with a gas/liquid separator, and the methane is reduced to carbon and hydrogen in an expendable CFR cartridge. Two such reactors are required to maintain continuous operation and allow for cartridge replacement.



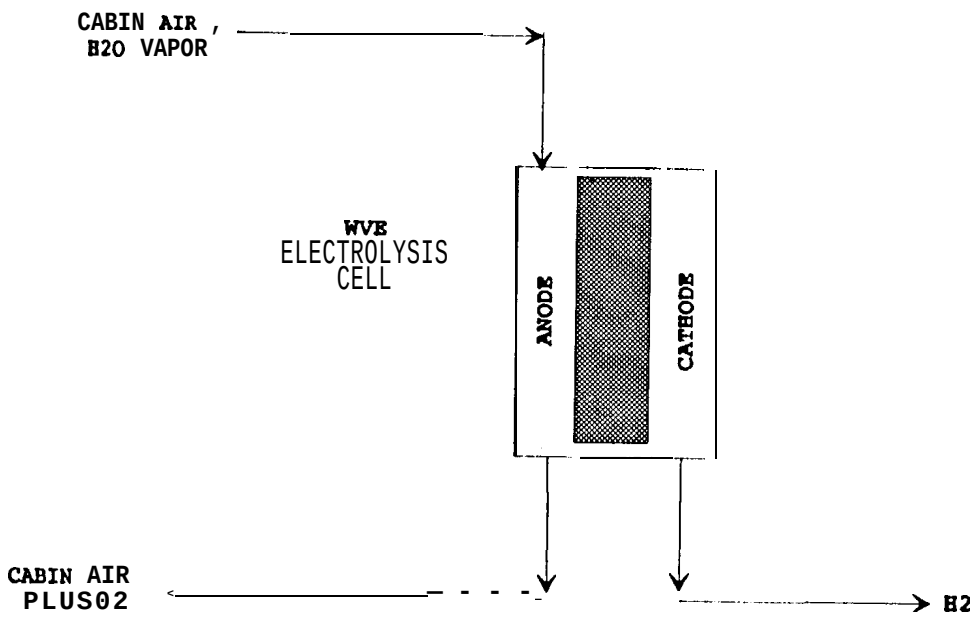
Process Flow Schematic for CO_2 Electrolysis/Boudouard

The CO_2 Electrolysis/Boudouard (CO_2EL/BD) process combines two SFE functions: CO_2 reduction and 0_2 generation. CO_2 is electrolyzed using a solid oxide electrolyzer producing O_2 and CO; CO is then catalytically decomposed into solid carbon and CO_2 via the Boudouard reaction; CO_2 is recycled back to the electrolyzer. Since this process generates O_2 directly from CO_2, thereby reducing (or eliminating) the oxygen generation via water electrolysis, the need to clean condensate for water electrolysis can be reduced also.



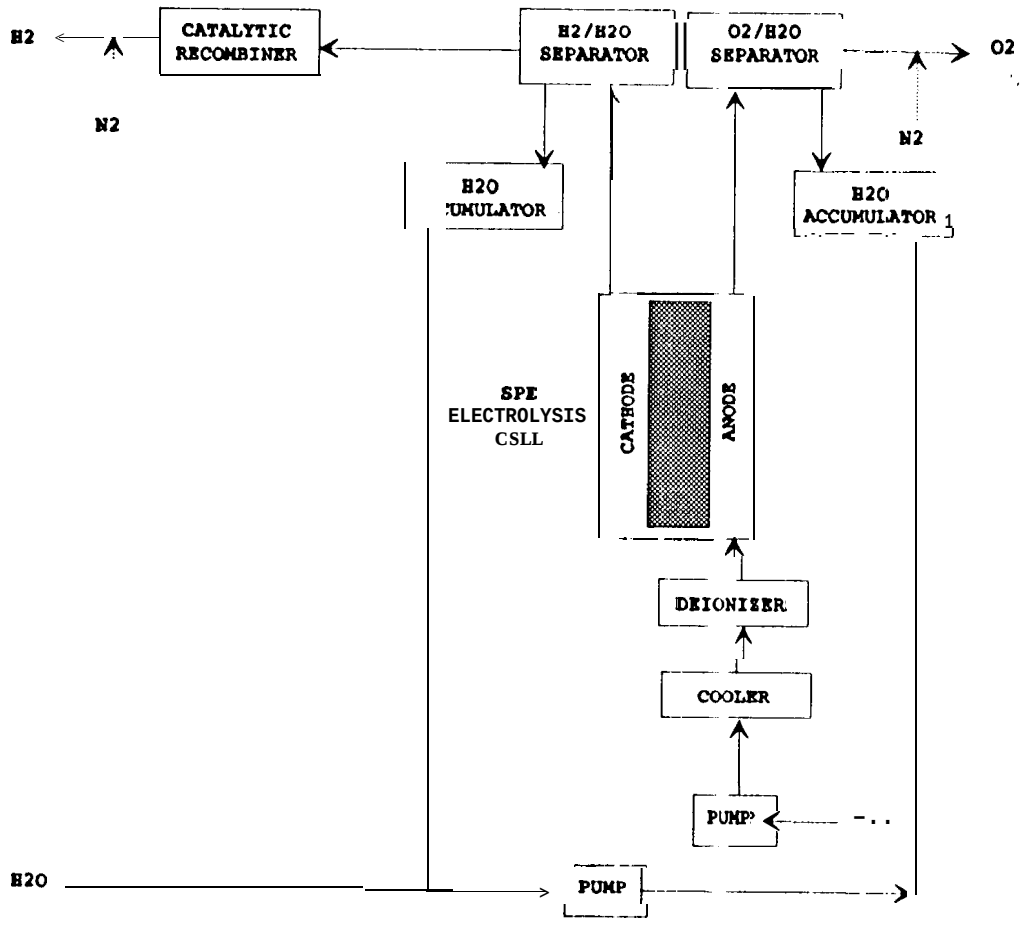
Process Flow Schematic for Static Feed Water Electrolysis

The Static Feed Water Electrolysis (SFWE) process electrolytes water to produce H_2 and O_2. Water is fed to the feed compartment where it diffuses as a vapor through the water feed membrane and into the anode. The cell electrolyte is an aqueous KOH held on a retention matrix. H_2 and O_2 are generated in the cathode and anode, respectively. N_2 is used for purging and pressurization purposes. Normal operating conditions are 80°C and 12 atm.



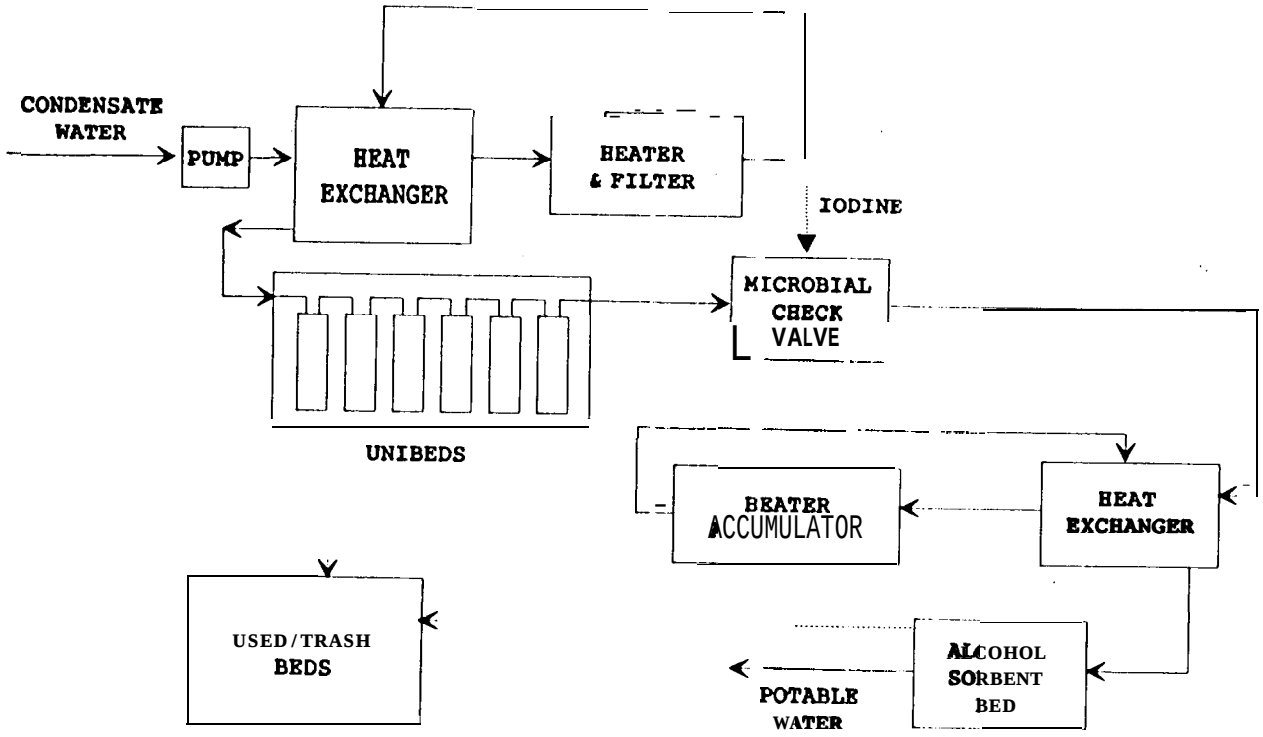
Process Flow Schematic for Water Vapor Electrolysis

The Water Vapor Electrolysis (WVE) uses a hydroscopic electrolyte (H_2SO_4) to absorb H_2O vapor from the cabin air and generate O_2, H+ions, and electrons in the anode compartment. At the cathode, H+ ions are joined with electrons to generate H_2.



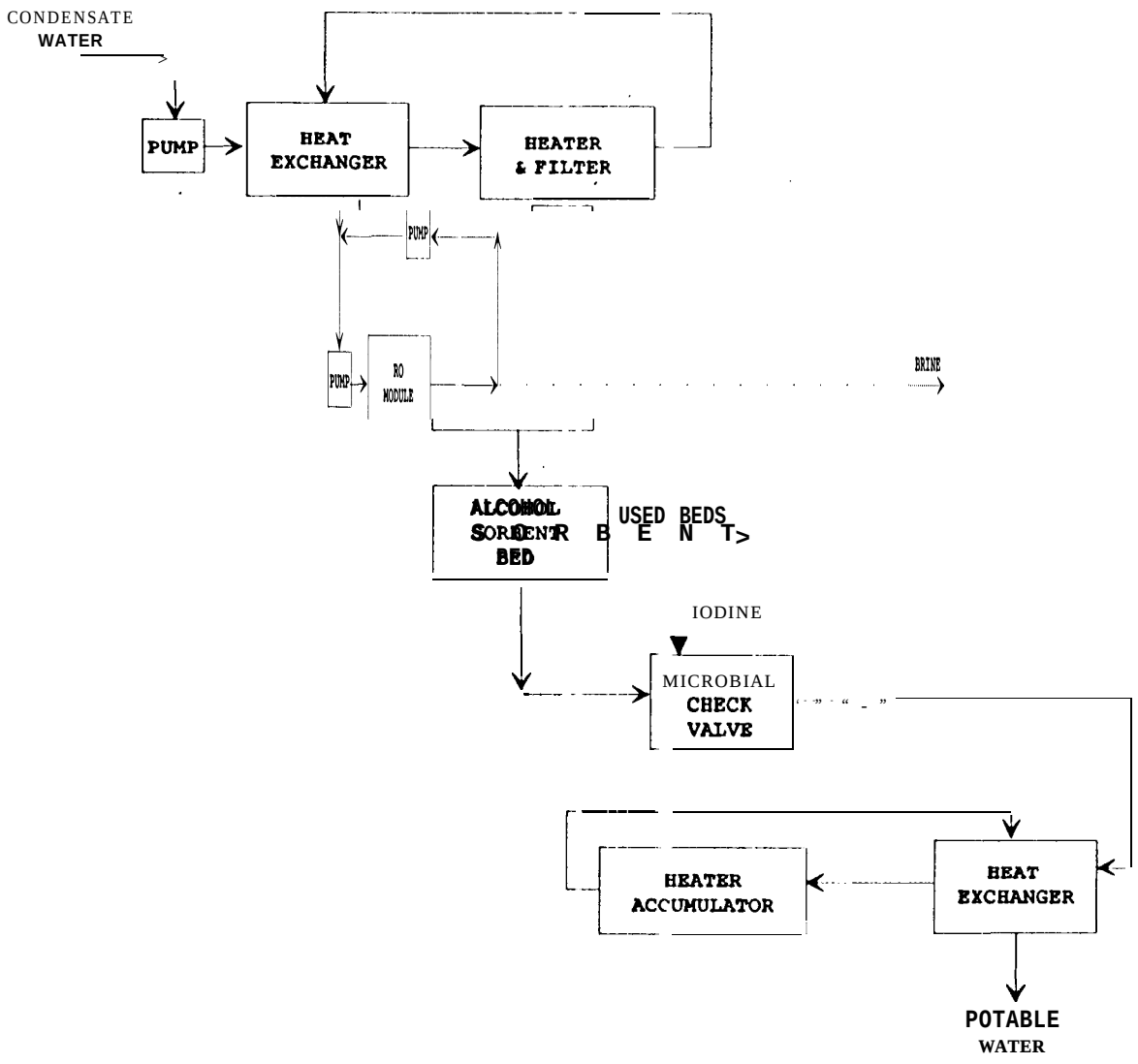
Process Flow Schematic for Solid Polymer Electrolyte

The Solid Polymer Electrolyte (SPE) uses a membrane made of sulfonated perfluoro-linear polymer (NAFION) . When fully hydrated with H_2O, the membrane is an excellent conductor and functions as the electrolyte. Deionized and cooled H_2O is fed to the anode where it is decomposed to 0_2, H+ ions, and electrons. The electrons travel through the external electrical circuit to the cathode, while the H+ ions migrate from anode to cathode by passing between the fixed, hydrated sulfonic acid groups. The H+ ions and electrons recombine on the cathode to evolve as H_2. Both H_2 and O_2 evolved gases contain water droplets that are separated from the gas phase. The recovered liquid H_2O is returned to the anode from H_2O accumulators. A recombine catalytically reacts 0_2 in the H_2 that may occur due to 0_2 to H_2 cross-leakage. The SPE cell operates at 500°C and 14 atm on the O_2 side; the H_2 side is at a lower pressure than the O_2 side. N_2 is provided to maintain O_2 pressures above H_2 pressure and for purging purposes.



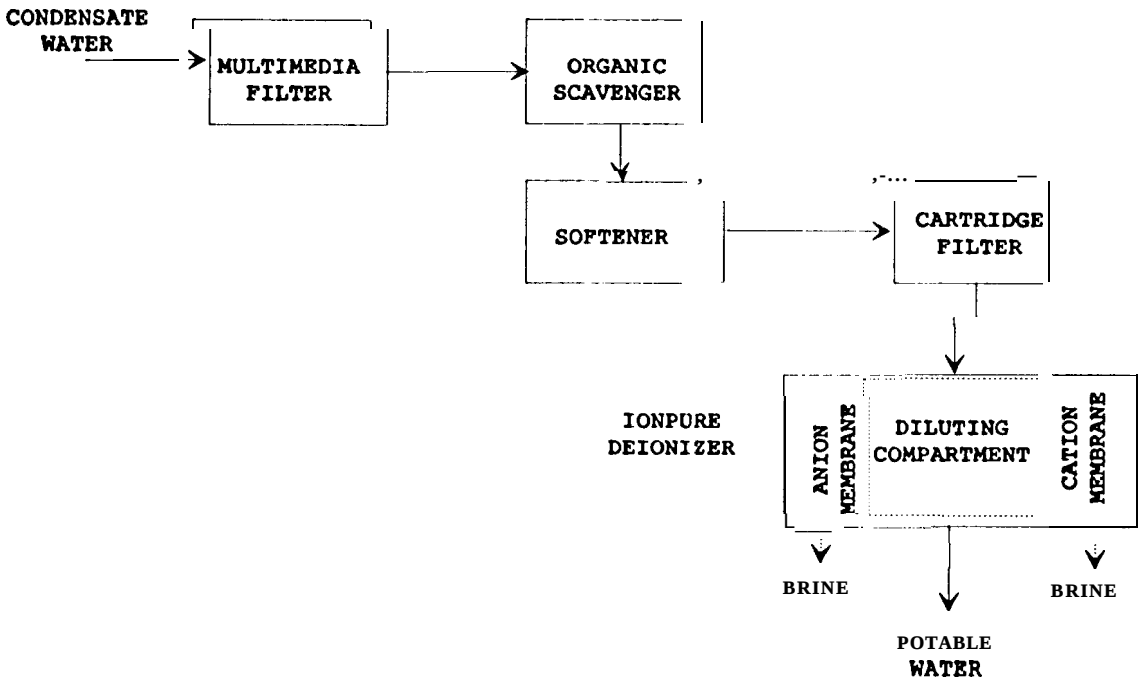
Process Flow" Schematic for Multi filtration for Potable Water Processing

The Multifiltration System is designed to produce potable quality water using expendable adsorption beds to remove both dissolved and ionic impurities. Water entering the process is first heated to 125°C and sterilized for 40 minutes; it is also filtered to remove any bacteria and particulate present. Flow is then directed to a series of six unibeds composed of an adsorption bed containing activated carbon and an ionic exchange resin bed operating at 25° to 45°C; the goal is to have an effluent with a total organic carbon concentration of 500 ppb or less. Eventually, the first bed reaches storage capacity and is removed. The remaining beds are moved up to fill the gap, and a fresh bed is placed at the end of the series. Microbial growth is impeded by heating and chemically treating the processed water at similar temperatures and residence times as the first heater/filter. Downstream of the unibeds iodine is injected into the process stream . The stream is then passed through an alcohol sorbent bed for the purpose of removing low molecular weight alcohols.



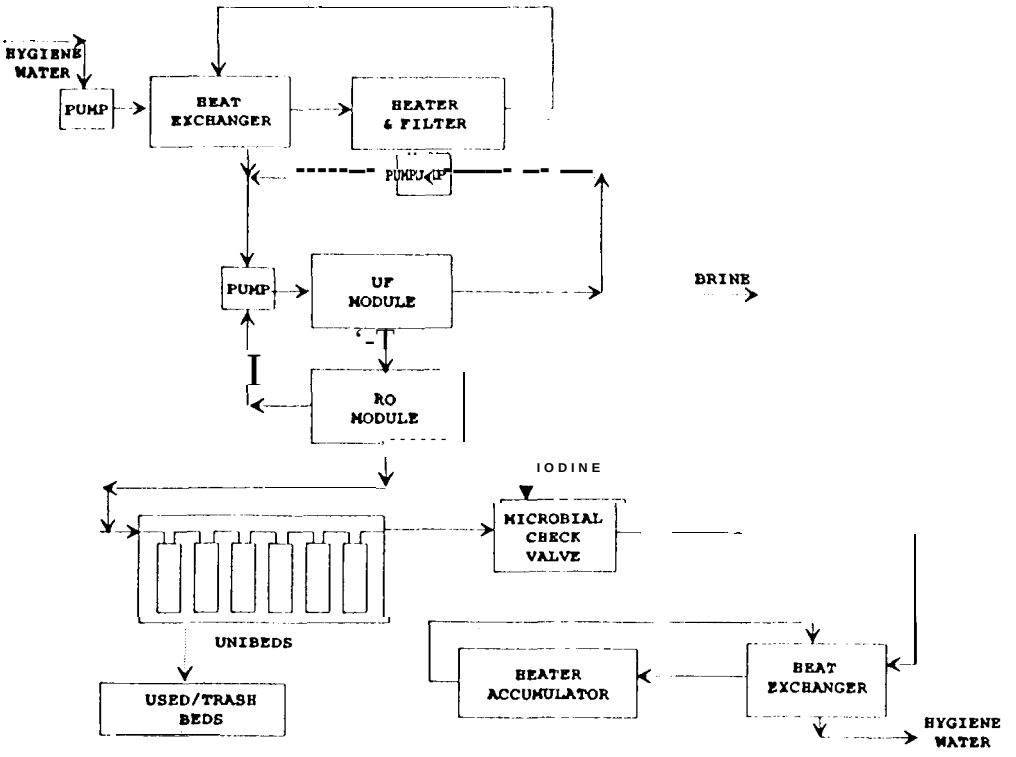
Process Flow Schematic for Reverse Osmosis for Potable Water Processing

The Reverse Osmosis (RO) process for potable water processing is designed to remove both dissolved and ionic impurities. Water entering the process is first heated to 125°C and sterilized for 40 minutes; it is also filtered to remove any bacteria and particulate present. Flow is then directed to an RO module that operates at 13 atm and 45°C. Brine is flushed from the system several times per day. The permeate is passed through an alcohol sorbent bed used to remove low molecular weight alcohols. Microbial growth is impeded by heating and chemically treating the processed water at similar temperatures and residence times as the first heater/filter.



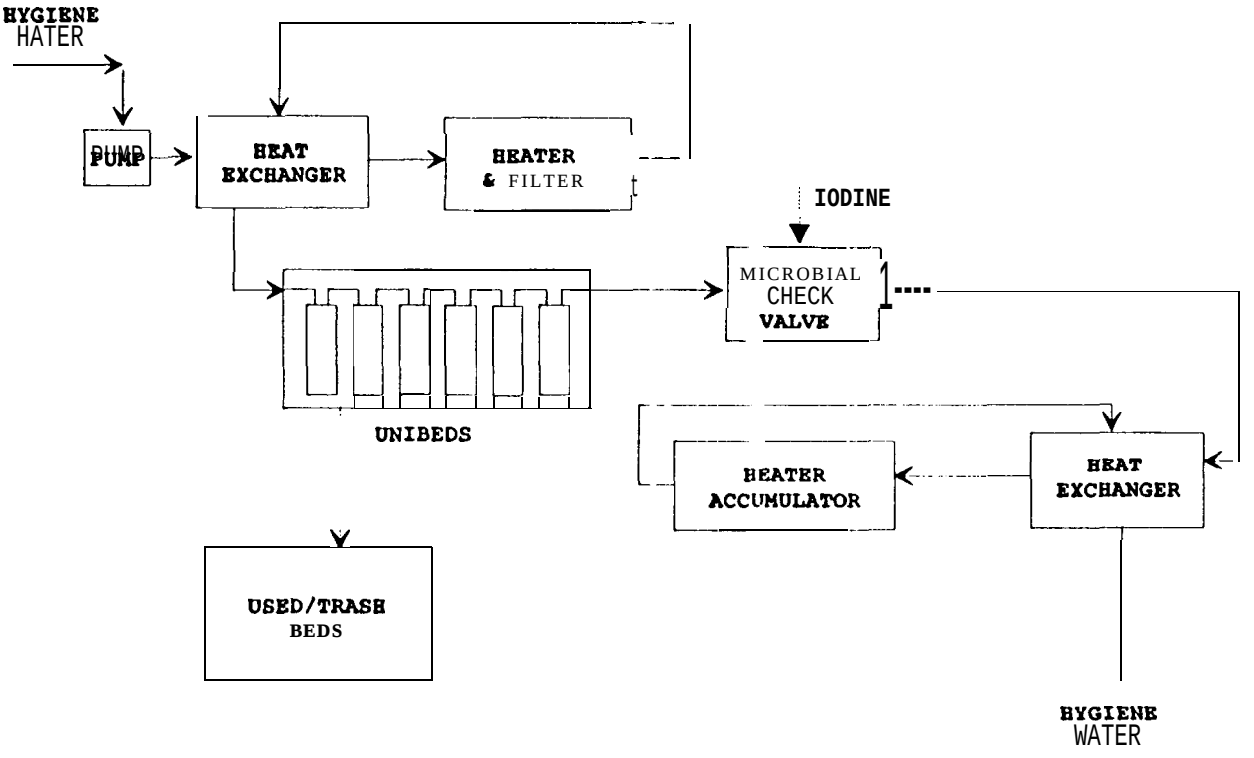
Process Flow Schematic for Electrochemical Deionization for Potable Water Processing

The Electrodeionization (ELDI) process utilizes ion exchange resins and membranes to deionize feed water. The ionpure deionizer contains ion exchange membranes that act as barriers to bulk water flow. The deionizer is divided into three adjacent compartments: a diluting compartment bordered on either side by a concentrating compartment. Feed water enters the diluting compartment (after pretreatment of the feed water by the multimedia filter, organic scavenger, and softener) , which is filled with the ion exchange resins, transferring through these resins in the direction of an electrical potential gradient applied across the compartments. Due to the semipermeability properties of the ion exchange membranes and the directionality of the potential gradient, ion concentration will decrease in the diluting compartment and increase in the concentrating compartments. The system outputs brine from the concentrating compartments and purified deionized water from the diluting compartment. The ion exchange resin is continually electrically regenerated.



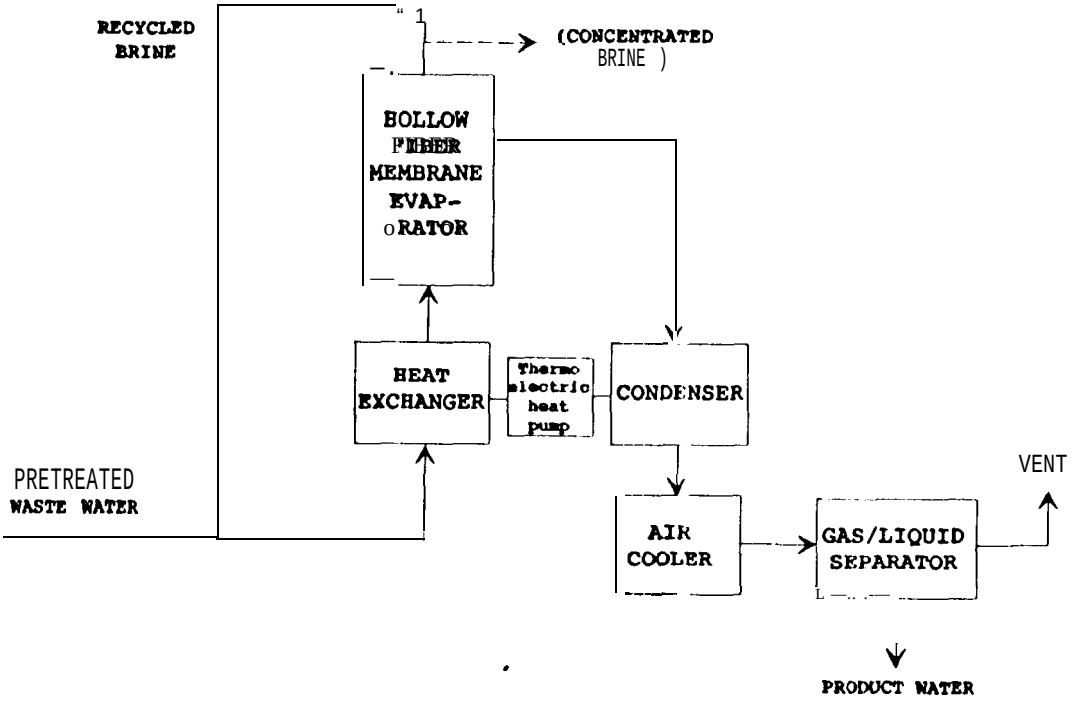
Process Flow Schematic for Reverse Osmosis for Hygiene Water Processing

The Reverse Osmosis (RO) is designed to produce hygiene quality water using a combination of an Ultrafiltration (UF) Module (to remove suspended solids, colloids, and macromolecules), a RO module (to remove salts and compounds with molecular weights >100) , and expendable adsorption beds to remove both dissolved and ionic impurities from the RO permeate. The process is similar to that used for potable water processing with the exception of the lack of alcohol sorbent beds, the addition of the UF Module, and the type of material in the Unibeds. Water entering the process is first heated to 125°C and sterilized for 40 minutes; it is also filtered to remove any bacteria and particulate present. Flow is pumped to the UF Module with UF permeate entering the RO module. Brines from UF and RO are recycled and purged periodically. Flow is then directed to a series of six unibeds composed of an adsorption bed containing activated carbon and an ionic exchange resin bed operating at. 25° to 45°C; the goal is to have the effluent reach a total organic carbon concentration of less than 10 ppm. Eventually, the first bed reaches storage capacity and is removed. The remaining beds are moved up to fill the gap, and a fresh bed is placed at the end of the series. Microbial growth is impeded by heating and chemically treating the processed water at similar temperatures and residence times as the first heater/filter. Downstream of the unibeds, iodine is injected into the process stream.



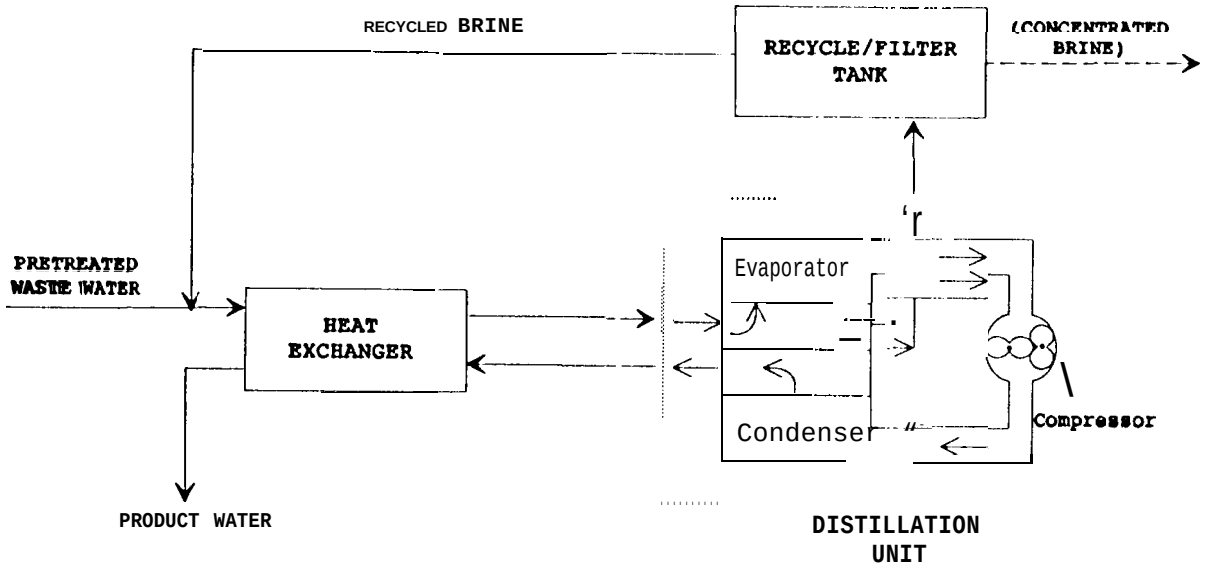
Process Flow Schematic for Multifiltration for Hygiene Water Processing

The Multifiltration System is designed to produce hygiene quality water using expendable adsorption beds to remove both dissolved and ionic impurities. The process is similar to that used for potable water processing with the exception of the lack of alcohol sorbent beds and the type of material in the unibeds. Water entering the process is first heated to 125°C and sterilized for 40 minutes; it is also filtered to remove any bacteria and particulate present. Flow is then directed to a series of six unibeds composed of an adsorption bed containing activated carbon and an ionic exchange resin bed operating at 25° to 45°C; the goal is to have the effluent reach a total organic carbon concentration of less than 10 ppm. Eventually, the first bed reaches storage capacity and is removed. The remaining beds are moved up to fill the gap, and a fresh bed is placed at the end of the series. Microbial growth is impeded by heating and chemically treating the processed water at similar temperatures and residence times as the first heater/filter. Downstream of the unibeds iodine is injected into the process stream. The stream is then passed through an alcohol sorbent bed for the purpose of removing low-molecular-weight alcohols.



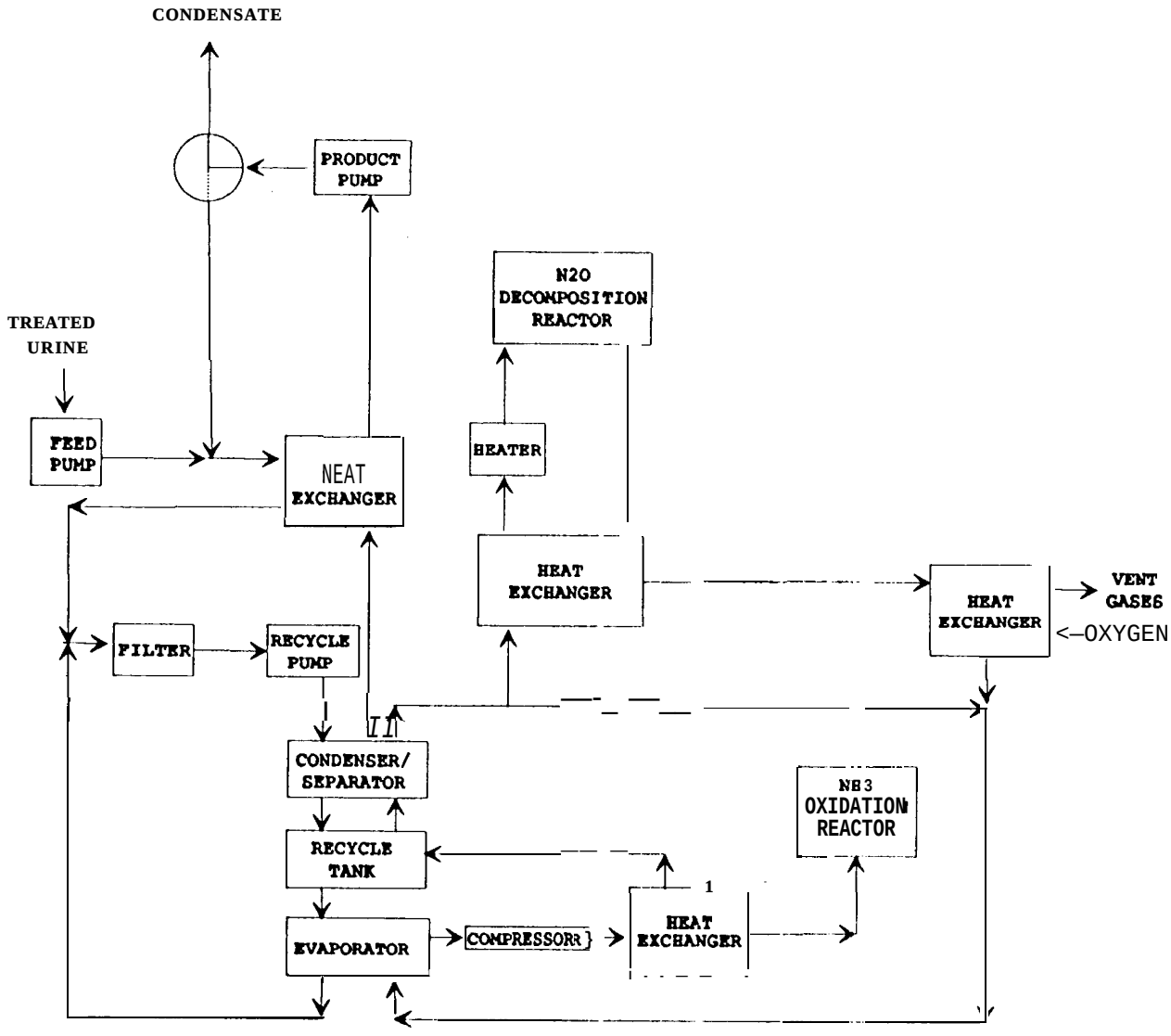
Process Flow Schematic for Thermoelectric Integrated Membrane Evaporation Subsystem

The Thermoelectric Integrated Membrane Evaporation Subsystem (TIMES) is designed to produce hygiene quality water from urine waste water attaining a 95% water recovery efficiency. Before entering TIMES, urine is chemically pretreated to fix free ammonia. After pretreatment, the waste water stream is first heated and then passed through hollow fiber membranes for evaporation at low temperatures. The evaporator consists of six bundles of 100 Nafion tubes each. Steam evaporates from the outer surface of the membranes and is partially condensed before flowing to an air cooled heat exchanger. Noncondensible gases entrained in the condensate stream are removed by a pump which functions as a gas/liquid separator. Unevaporated waste water is recycled until solid concentrations reach a predetermined level, at which time the concentrated brine is removed for disposal. Using thermoelectric devices, the latent heat of condensation is recovered and reused in the evaporation process.



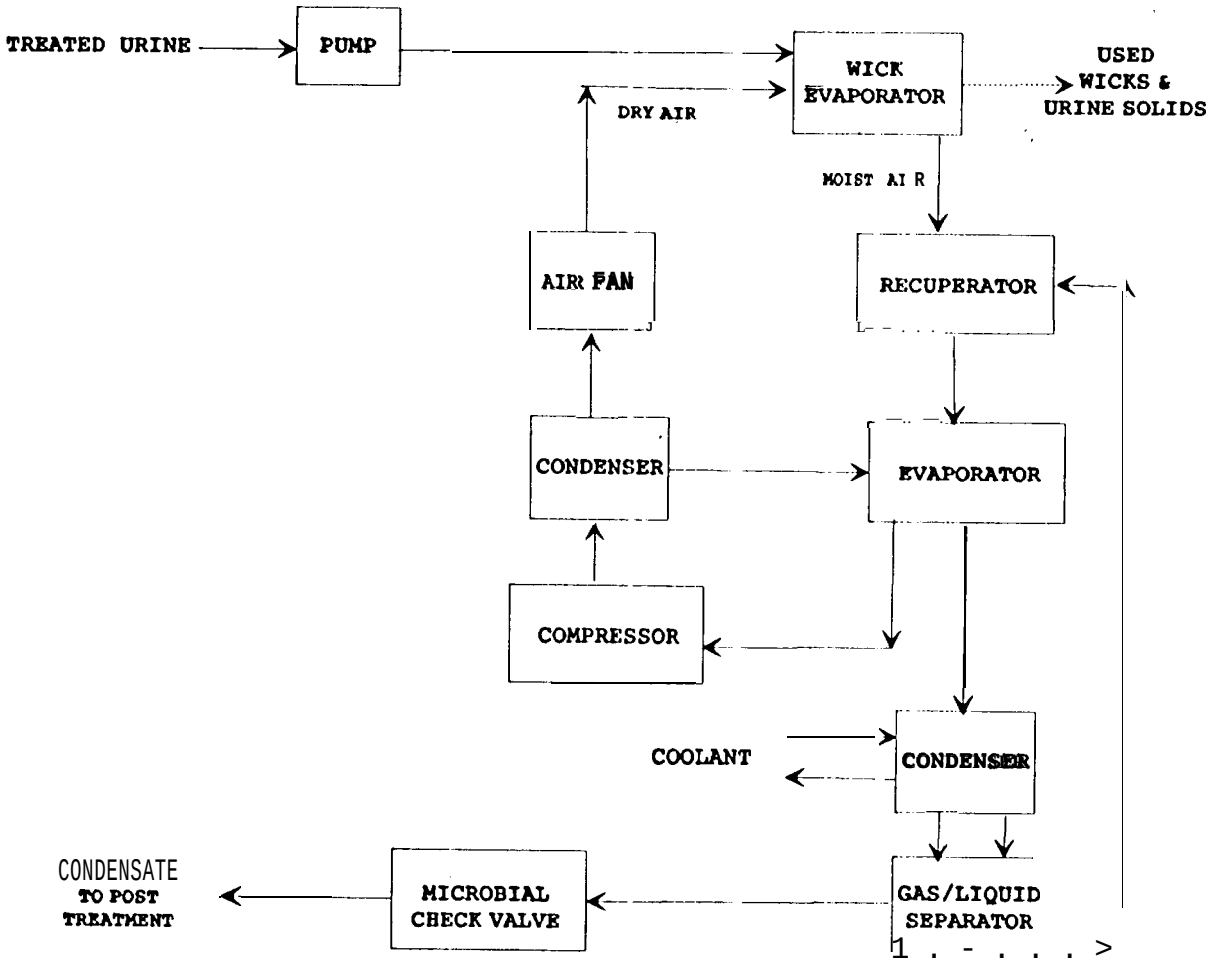
Process Flow Schematic for Vapor Compression Distillation

The Vapor Compression Distillation (VCD) system maintains a vapor/liquid interface using centrifugal force created by a rotating drum. Waste water is discharged to the inner surface of a centrifugal evaporator drum inside the distillation unit. Water vapor is removed from the evaporator, compressed to raise its saturation temperature, and then forced against the outer surface of the rotating drum where it condenses. The latent heat of condensation is transferred through the drum wall and reused in the evaporation process. Unevaporated waste water is recirculated until solid concentrations reach a predetermined level, at which time the concentrated brine is removed for disposal.



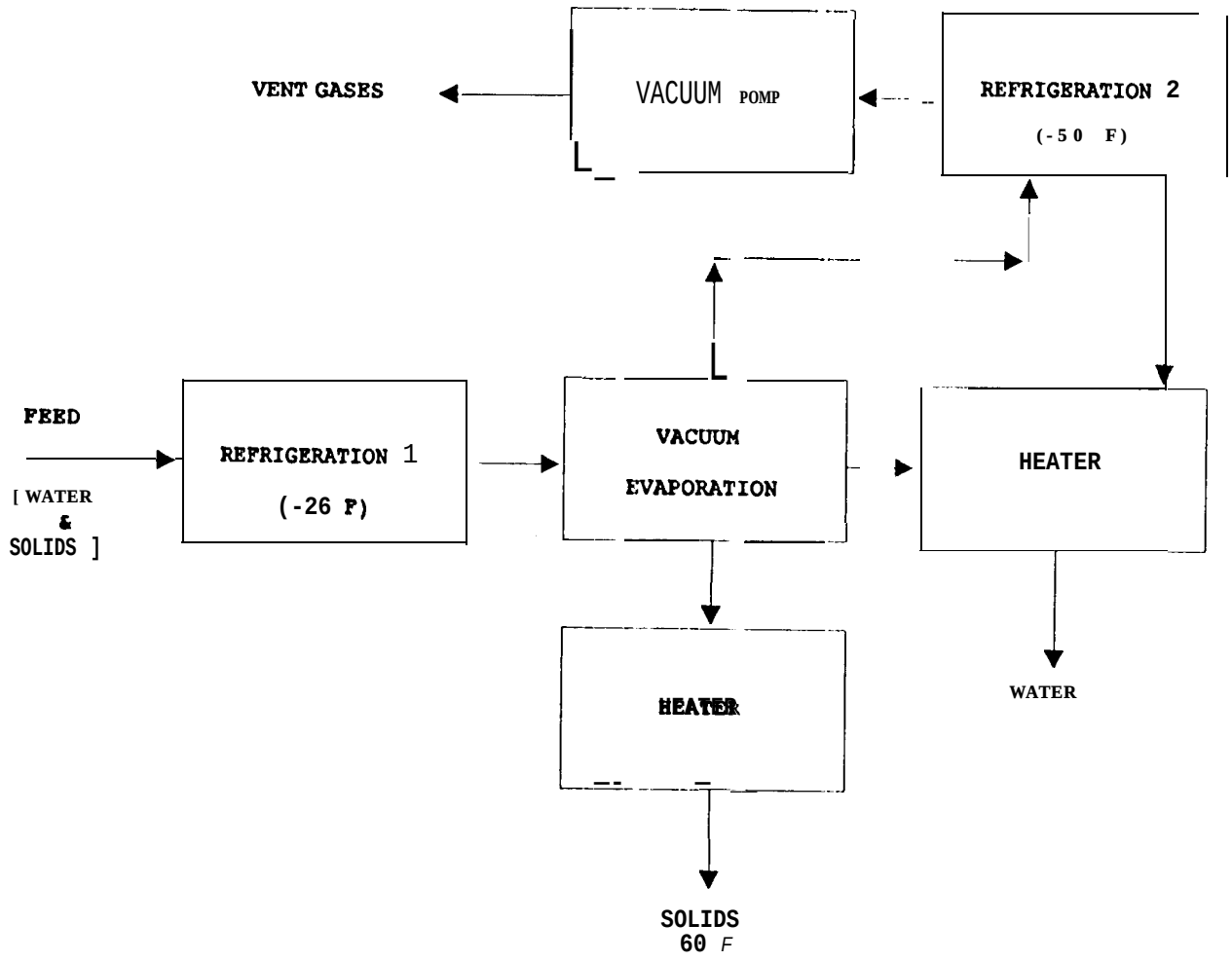
Process Flow Schematic for Vapor Phase Catalytic Ammonia Reduction for Urine Processing

The Vapor Phase Catalytic Ammonia Removal (VPCAR) Process utilizes catalytic reactors to react vaporized impurities in the feed water to innocuous gases. Urine is vaporized at 100″C in an evaporator. The process employs two catalytic reactors. The NH_3 oxidation reactor uses a Pt catalyst to oxidize NH_3 to a mixture of N_2 and N_2O and volatile organic hydrocarbons are oxidized to CO_2 and water vapor at 250°C. The N_2O decomposition reactor uses a Ru catalyst at 400°C to N_2 and 0_2. The recovered H_2O has little NH_3, few hydrocarbons, low conductivity, and only requires pH adjustment to be a candidate for potable water.



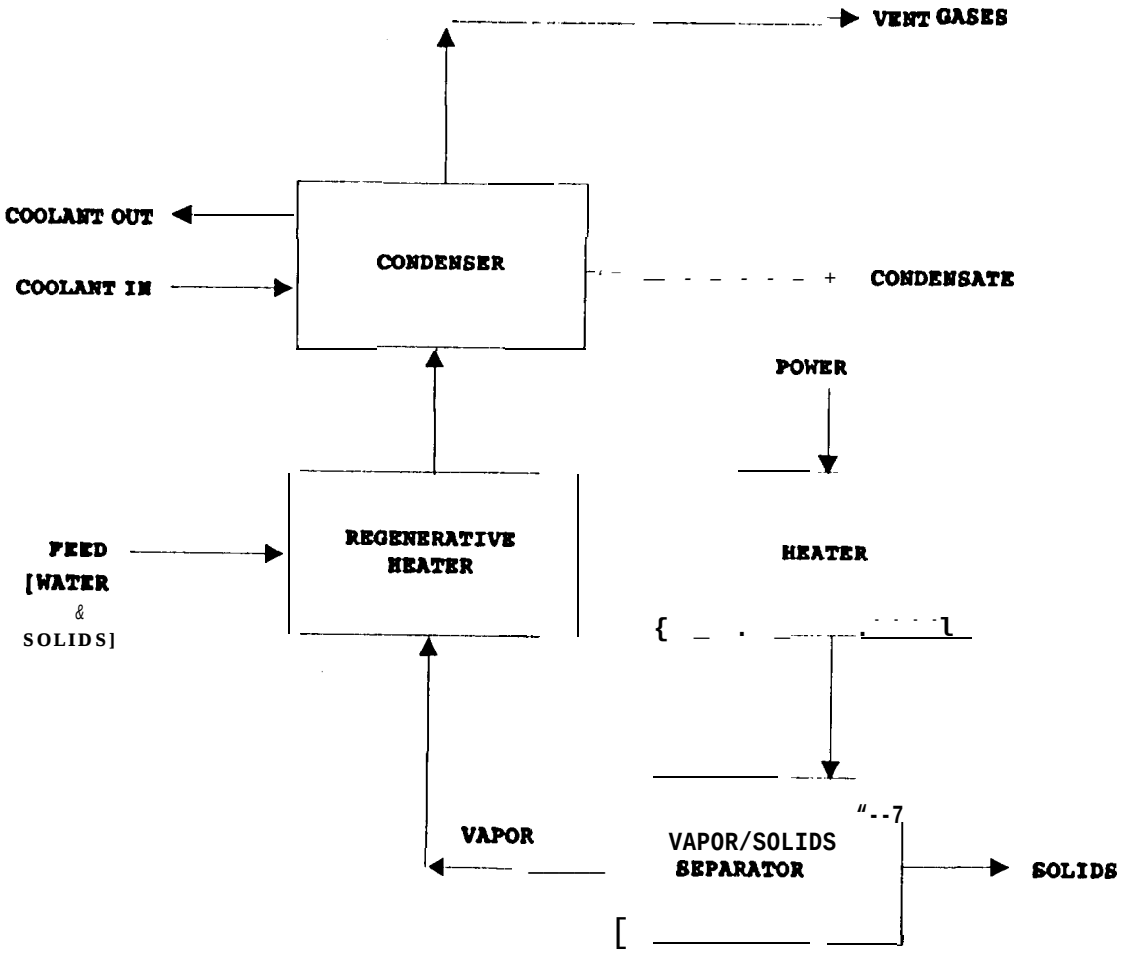
Process Flow Schematic for Air Evaporation for Urine Processing

In the Air Evaporation (AIRE) process, treated urine is pumped to a wick package along with a dry air stream. The circulating heated air evaporates water from the urine leaving solids in the wicks. When sufficient solids accumulate in the wicks, the feed is stopped and the loaded wicks are dried down and replaced. Humid air leaving the wick evaporator passes through a heat recuperator and a condensing heat exchanger. A water separator downstream of the condenser removes water from the air and pumps it out as condensate. Iodine is added to the water before it is sent to post treatment before it can be used as hygiene water.



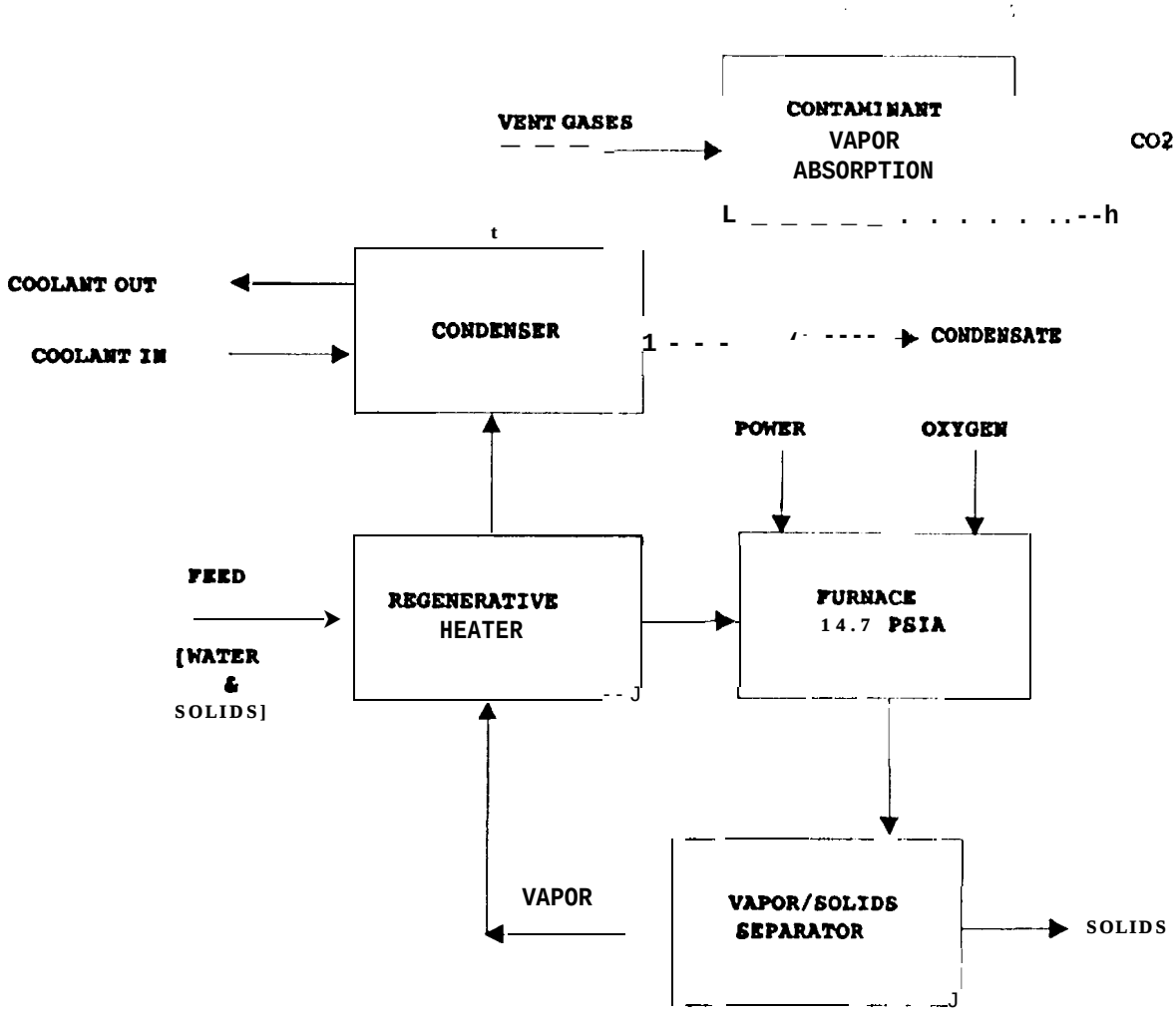
Process Flow Schematic for Freeze Drying

The freeze drying (FRZ) process consists of four major steps as illustrated above:
(1) pre-freezing at -3°C to freeze dissolved and suspended materials along with water;
(2) vacuum evaporation or sublimation of the frozen ice at <0.0001 atm;
(3) re-freezing water vapor at -15°C and
(4) melting of the frozen ice at 16°C.



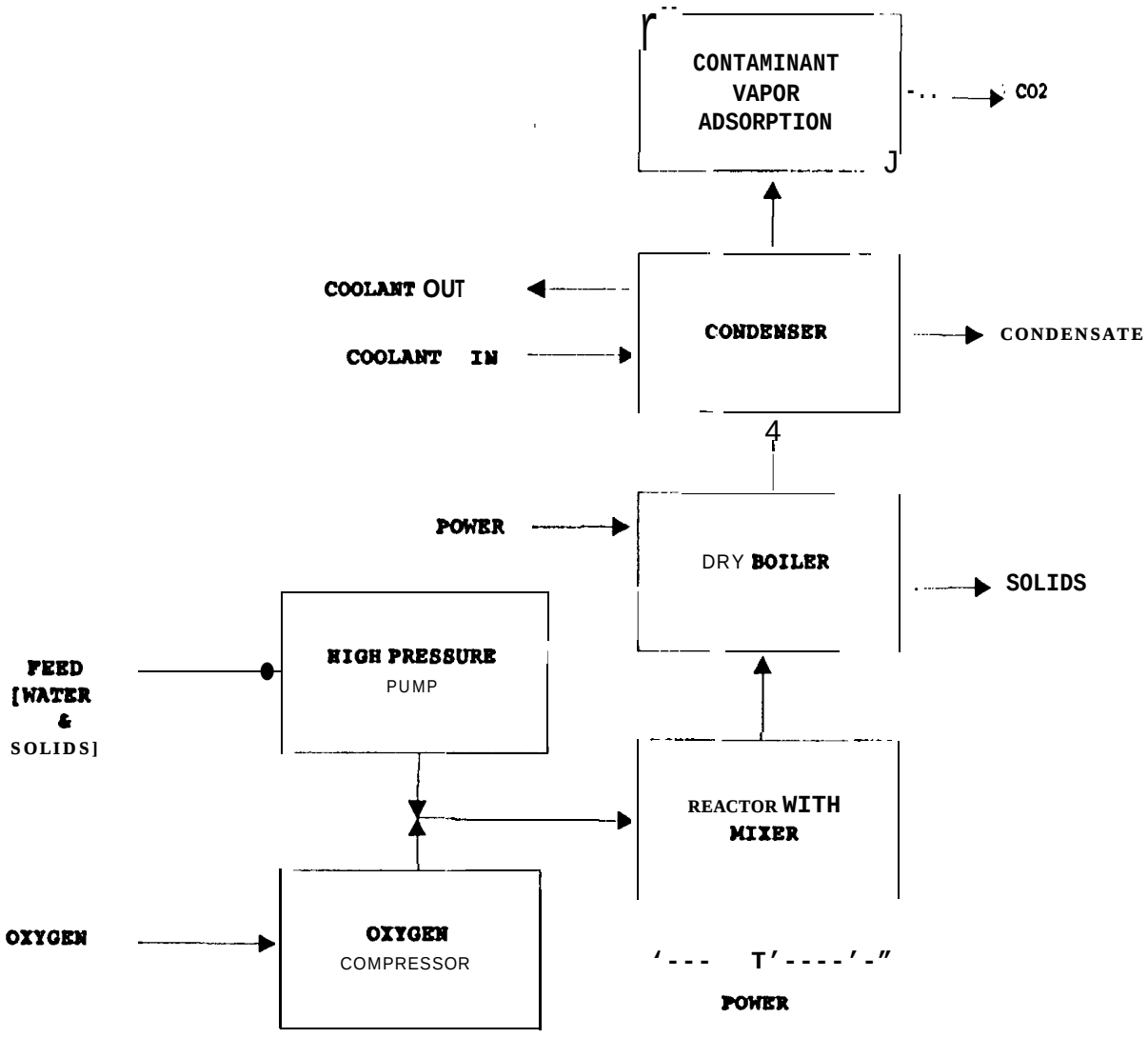
Process Flow Schematic for Thermal Drying

The Thermal (or hot) Drying (HD) process uses power to dry the feed at temperatures exceeding 150°C. Regenerative heaters are provided to increase the thermal efficiency. Potential waste heat sources, rather than electrical power, could be process waste heat from other physical/chemical processing steps, such as CO_2 reduction.



Process Flow Schematic for Combustion Oxidation

The Combustion ("COMB) Oxidation process uses pure oxygen to Incinerate the organics in the feed. Power is also required as the stream has a low heating value. An ambient pressure furnace is used; ash solids residue. is separated after incineration. After recovery of some of the waste heat in a regenerative heater, the water condensate formed from the original water and the oxidized organics is condensed. Unreacted or partially oxidized organics and other contaminant vapors are absorbed. CO_2 formed from oxidizing the organics is recycled totheair revitalization subsystem to reduce the CO_2 to carbon and oxygen.



Pr cess Flow Schematic for Wet Oxidation

The wet oxidation (WOX) process uses pure oxygen to oxidize the organics in the feed in a reactor maintained at 290°C atm and 150 atm. Power is also required as the stream has a low heating value; in addition, power is required to pump the feed waste stream and compress the oxygen. Ash solids residue is separated after the reactor in a dry boiler, operated at low pressure and over 230°C. The water condensate formed from the original water and the oxidized organics is condensed. Unreacted or partially oxidized organics and other contaminant vapors are absorbed. $C0_2$ formed from oxidizing the organics is recycled to the air revitalization subsystem to reduce the CO_2 to carbon and water.